# Design Flow and Implementation of an IoT Smart Power Socket

Carlos Mateo, Fernando Almagro, Won-Jae Yi and Jafar Saniie
*Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)*
*Department of Electrical and Computer Engineering*
*Illinois Institute of Technology, Chicago IL, U.S.A.*

*Abstract*— **This paper presents the design and implementation of an IoT smart power socket. Its main function is to provide services such as remote control, electrical protection, task automation, and monitoring through mobile applications using cloud services. In addition, this IoT smart power socket is supported by advanced software tools for time scheduling, grouping power sockets, and their operation managements, and analyzing power usage for electric cost optimization. Our system can be used in different domains including smart and remote access home automation systems, factory automation systems, security systems, and power usage management systems. Prototype of our power sockets is designed with locally-controlled sensors and actuators, exchanging information with a common central hub using the Bluetooth protocol to receive commands and send information to the user via the Internet connection.**

***Keywords—IoT, Smart Power Socket, Bluetooth Mesh Configuration, Remote Control API***

## I. Introduction

It is commonly known as a 'smart plug', a male/female electric plug and socket, which can be placed between the domestic electric network and the desired electronic device, offering advanced functions to control and monitor the related electronic appliance. These features may vary depending on the commercial product, ranging from remote monitoring of the power consumption to the control of the appliance status (ON or OFF). Such a device along with the IoT technology adaptation improves the quality of life, although, some old appliances may not join the IoT network due to their lack of 'smartness'. Solutions like the smart plugs will allow to include these appliances into the IoT network, bringing them 'smartness'.

Existing commercial solutions [1–6] are expensive, do not permit easy manual control, do not protect the electric appliance, and do not offer the chance to perform an automatic operation of the appliance. Given these drawbacks, a market niche is found in which a more affordable and intelligent smart plug may succeed. Furthermore, previous works on this matter [7–12] focus on energy management and not to enable the user to perform a smart and automated use of the appliances or provide advanced electrical protection.

Attending to these facts, we propose a smart plug product called 'Smart Power Socket' (SS), which shall match the highlighted unsupplied offer. The proposed SSs will include the following features:

- Remote control of the appliance via a mobile app
- Real-time monitoring of the SS status (ON/OFF)
- Real-time monitoring of consumed power
- Automatic operation based on ambient conditions (illuminance, temperature, humidity)
- Comfortable manual operation
- Schedules and groups creation
- Electrical safety
- Ability to interact with other SSs directly

The system to realize the proposed tasks with the aforementioned features is shown in Figure 1. Each SS is equipped with a local microprocessor in charge of managing the information flow and trigger the socket's power switch. All local SSs connect to the same SS Hub via Bluetooth, sending and receiving information of their current state via the user's commands, hence creating a wireless sensors' network (WSN) for all SSs [13]. To receive the user's commands from a cloud server, each SS Hub may connect to the Internet access and redirect them all to the SSs. In this paper, we introduce a prototype, as a proof of concept, realizing the basic tasks as described above.

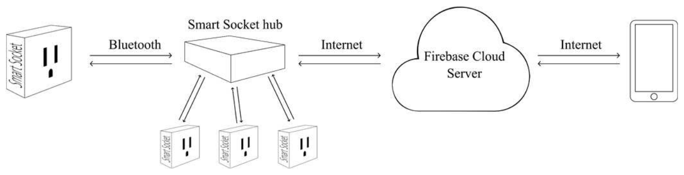


Figure 1. Overview of Distributed Smart Power Socket Architecture

## II. System Design

### A. SS Local Software Design

The general software design of the local SS shall perform the mentioned tasks as well as been implementable in a generic microprocessor platform. To properly define the current state of the SS, the algorithm considers the following state variables:

- **Mode:** indicates the working mode of the SS: manual and automatic operation based on temperature, humidity or illuminance.
- **Threshold**: The SS state switches in automatic operator mode comparing the variable's value with *threshold*.
- **Up/Down**: Indicates when to close the power switch with respect to the *threshold* in the automatic operation modes. If *Up/Down* is True, the switch will be closed only if the measurements are above threshold.
- **Electrical Safety**: If electrical conditions are unsuitable (overvoltage, undervoltage, overload), SS will remain OFF.

- **External trigger**: User's remote action over the SS. A flank in this variable will invert the SS state.
- **Manual block**: A direct user operation on the SS status blocks it, being only released after a consecutive action, avoiding any state switch due to external conditions but *electrical safety*. It is considered a direct user action a flank in the external trigger and detecting a waving hand in front of the SS.

The aforementioned state-definitory variables will acquire a given value in each local SS, depending on received inputs (measurements, commands) and last status. This process is explained in the flowchart shown in Figure 2.

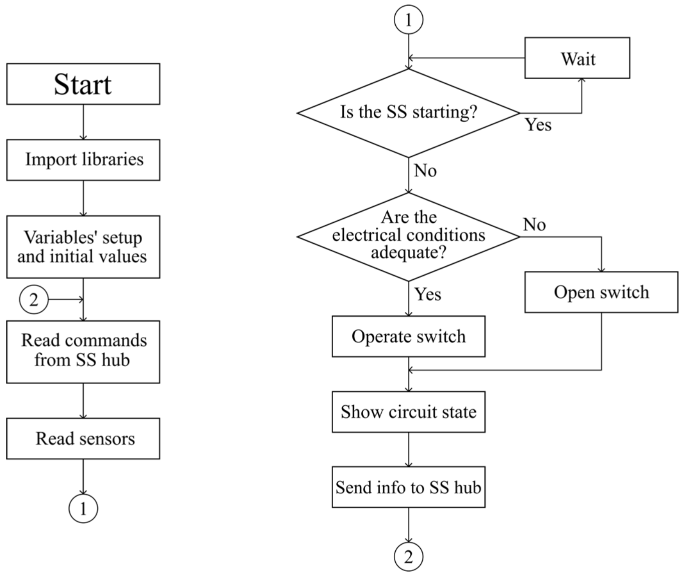


Figure 2. Flowchart of Local Smart Power Socket Main Function

Given the risk factor of the tasks which this algorithm controls (closing the switch by error may lead to material and human damage), several measurements have been adopted to prevent errors and potential damage:

- **Bluetooth Message Format**: A specific message format has been adopted, rejecting any information not compliant with it.
- **Received Message Consistency**: Only messages received with consistency will be followed.
- **Default Values**: Local SS has default values ensuring a released open state for the user to manage it at will.
- **Sensors' Noise**: Rolling average is implemented to avoid a fake switch of the SS state.
- **Electrical Safety**: As mentioned before, voltage and current measurements will be used to open the switch in case that inadequate conditions are found.

## *B. SS Hub Software Design*

Analogously, the SS hub realizes its part for the global system depending on the described features. This software is to be implemented in a more powerful environment than the local SS, able to establish Bluetooth and Internet connections.

The SS hub's main function works in a loop-based operation as shown in Figure 3. Initially, the SS hub will pair and connect to all SSs found within range (previously specified by the user), then sending commands and receiving information from each of them on a rotational basis. Once the information has been exchanged for a single SS, the data exchange is established with the user and recorded commands through the Internet.

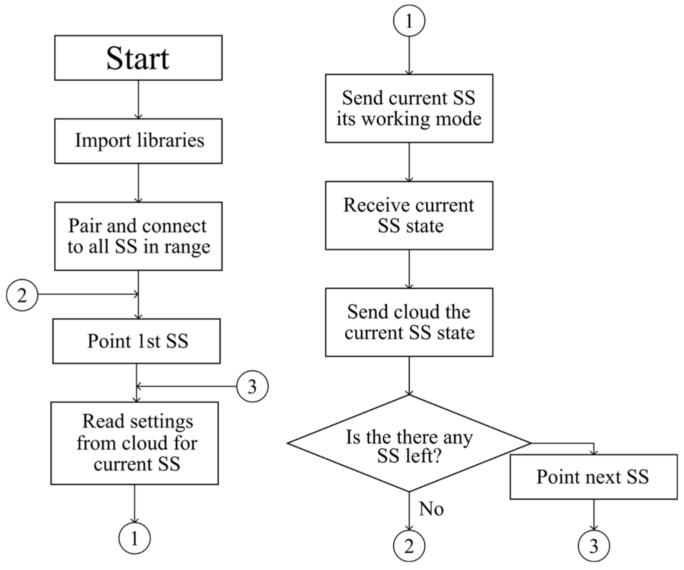

Start
Import libraries
Pair and connect to all SS in range
2
Point 1st SS
3
Read settings from cloud for current SS
1
1
Send current SS its working mode
Receive current SS state
Send cloud the current SS state
Is the there any SS left?
No
Point next SS
2
3

Figure 3. Flowchart of Smart Power Socket Hub Main Function

Concerning the security of the system, a similar approach to the local SSs has been adopted including message consistency and format checks. Also, to establish communication between the SS hub and the Internet, a unique ID for each SS hub is required. If the unique ID does not match the cloud's stored data, the communication will not be established. This prevents the intrusion of any unwanted external elements in the SS hub via the cloud communication link.

As shown in Figure 4, a Bluetooth-based meshing system can be adapted to our system design in which the local SSs can send commands to other SSs if they are out of range of the SS hub.

## *C. Hardware Design*

The hardware design completely depends on the platform of choice in which to develop the SS system, but it shall ensure the proper development of the specified tasks. A prototype of the hardware design is described in the next section.

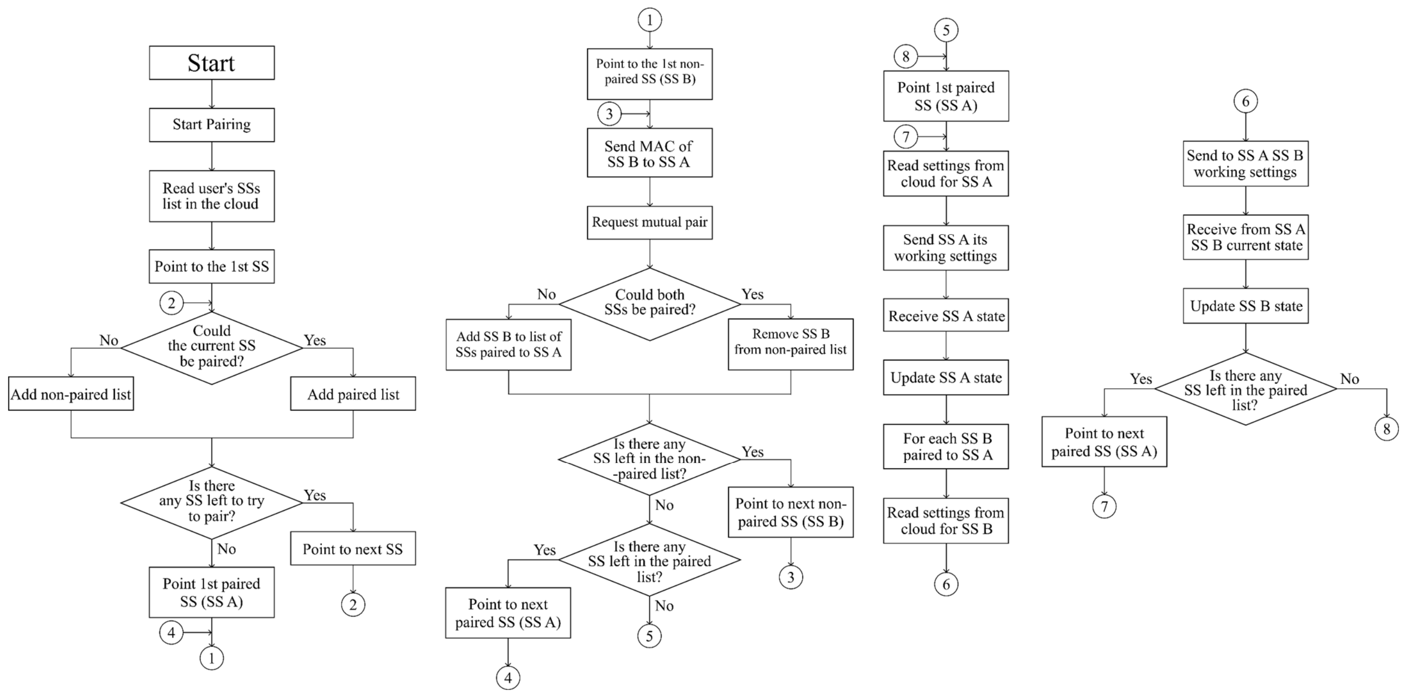

Start
Start Pairing
Read user's SSs list in the cloud
Point to the 1st SS
2
Could the current SS be paired?
No
Yes
Add non-paired list
Add paired list
Is there any SS left to try to pair?
Yes
No
Point to next SS
2
Point 1st paired SS (SS A)
4
1
1
Point to the 1st non-paired SS (SS B)
3
Send MAC of SS B to SS A
Request mutual pair
Could both SSs be paired?
No
Yes
Add SS B to list of SSs paired to SS A
Remove SS B from non-paired list
Is there any SS left in the non--paired list?
Yes
No
Point to next non-paired SS (SS B)
3
Is there any SS left in the paired list?
Yes
No
Point to next paired SS (SS A)
4
5
5
8
Point 1st paired SS (SS A)
7
Read settings from cloud for SS A
Send SS A its working settings
Receive SS A state
Update SS A state
For each SS B paired to SS A
Read settings from cloud for SS B
6
6
Send to SS A SS B working settings
Receive from SS A SS B current state
Update SS B state
Is there any SS left in the paired list?
Yes
No
Point to next paired SS (SS A)
7
8

Figure 4. Bluetooth Mesh Flowchart for Direct Communication among Smart Power Sockets

## III. Implementation

The implementation phase covers the development of the mentioned proof-of-concept prototype, which only for this purpose is based on Arduino and Raspberry Pi devices, to play the roles of local SS and SS hub respectively.

### A. *Hardware Implementation*

Regarding all desired features, the chosen microprocessor's performance, and market offer, the prototype hardware design shown in Figure 5 is proposed. In broad terms, one extra module has been included for each automated operation mode (temperature and humidity sensor, photoresistor, and ultrasound distance sensor) and an additional one to able the microprocessor to use the Bluetooth protocol. The main limitation of the mentioned Bluetooth transceiver is its inability to establish a meshed connection.

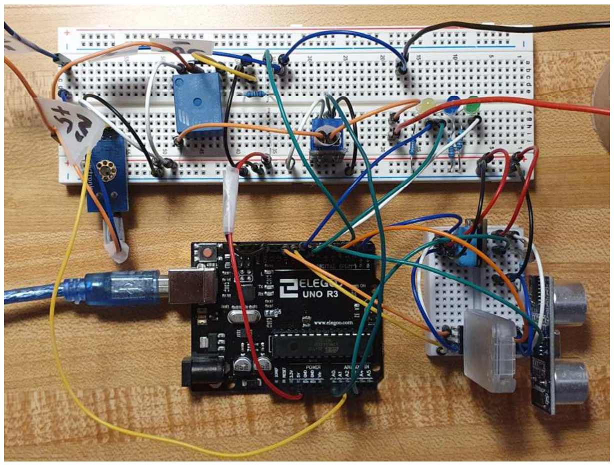


Figure 5. Smart Power Socket Prototype Hardware

The power input of the local SS comes from a power source adaptor attached in parallel to the actual socket's plug. To realize the prototype of the system and to reduce the electric hazard on the prototype, the SS is designed to handle a less-powerful power input (<25 VDC, 10 A) than the typical U.S. wall outlet standards, where the power will be provided by the SS hub platform. Also, a power consumption estimation has been performed, to assure that the microprocessor can handle all the modules' power demand. Three LEDs have been included to indicate the current state of the SS (electrical safety, blockage, and switch state) as shown in Figure 6 (next page).

### B. *Local SS and SS Hub Software Implementation*

Regarding the features of the chosen hardware platforms, the software is tailored to maximize system performance. For the local SS, the code is developed using C++ language, ensuring fast execution of the main algorithm, reacting promptly and reliably to the user's commands. The SS hub software is developed using Python, taking advantage of its versatility and flexibility when dealing with tasks of a different kind (Internet and Bluetooth communication, data processing). The whole SS hub software is divided into three main modules:

- **SS Hub Management:** Common nexus of all the modules. Manages the proper and ordered execution of the global SS hub algorithm. Oversees first pairing and connecting to all user-defined SSs in range and then starting the loop-based tasks. To boost the power of the hardware environment and programming language, SS hub – local SS and SS hub – cloud parallel processes are carried out taking advantage of the subprocess method. Information between parallel processes is exchanged via locally created CSV files.

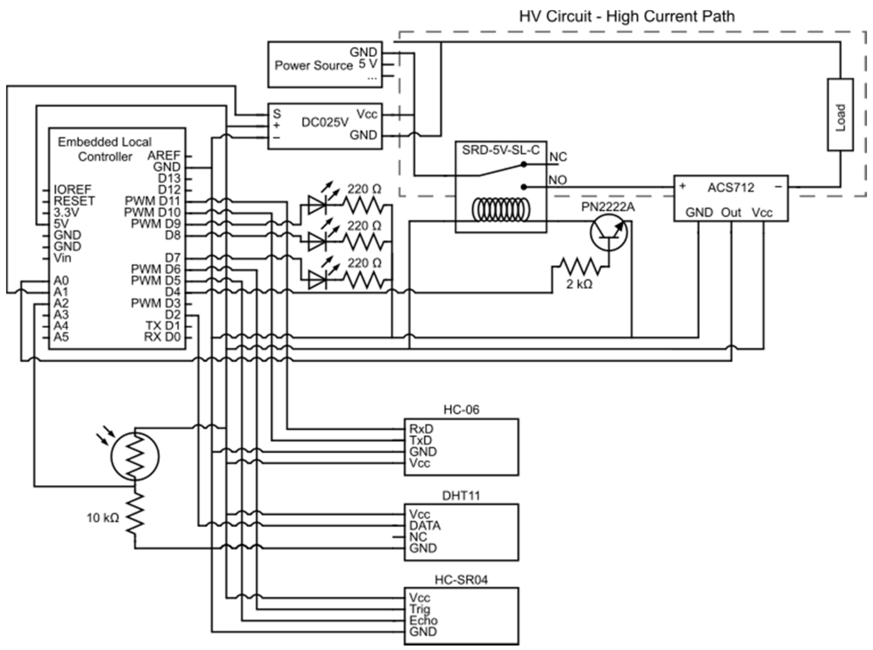


Figure 6. Hardware Schematic of the Smart Power Socket

- **SS local to SS hub:** Custom-created library which encapsulates functions to exchange information securely and reliably with the cloud and local SSs.
- **SS local to cloud**: Custom-created library, analog to the previous one, to exchange data with the cloud.

## C. *SS Cloud Implementation*

In terms of the backend, the concept of a serverless paradigm is introduced in this project. Nowadays, mobile app developers can make use of services or applications hosted by a third party. This is known as MBaaS (Mobile Backend as a Service). Some of the advantages of using MBaaS are that developers do not have to maintain the backend infrastructure, and they can optimize the cost since it costs only for the exact amount of resources that are being used, whereas they still have the benefits of autoscaling and high availability. In this paper, two MBaaS have been used: Firebase Authentication and Firebase Firestore. Firebase Authentication, an identity, and access management system, oversees identifying the users using a unique identification key assigned to each user when they are created. Firebase Firestore is the solution presented by Firebase for the database, a NoSQL database that consists of documents organized into collections. One document can have collections inside, as well as the fields of any type like Boolean, string, or number, but also maps or arrays. Making queries is intuitive, and the data can be updated and presented in real-time by using Snapshot Listener, code elements that can create connections between the app or script and the database, receiving real-time updates or notification when there are any changes in the database, either something new is added, modified or removed.

A diagram of the database architecture is shown in Figure 7. The black boxes represent the collections, whereas the grey ones are the documents. It can be observed that the names of the document are written between braces, that is because they are wildcards and that names are representing an actual “userID” or “ssMAC”. At the bottom of the diagram, there is the “users” collection, that each time that a user registration will be added to a new document, whose ID will be the “userID”. In that document, unique for each user, the user will be able to add documents by adding hubs, defining the hub code and the hub name, action that will create a new document in the hubs collection of the user, but also in the “main” hubs collection, in which each hub document will have three different collections, one for the groups, that will contain the document corresponding to each group including all the MACs of the hubs added to that group, and the “sockets_control” and “sockets_states” collections. In these two last-mentioned collections, each Smart Socket will have a document with its BT transceiver MAC as document ID. Examples of how the SS collections for control and state of the SSs are shown in Figure 8.

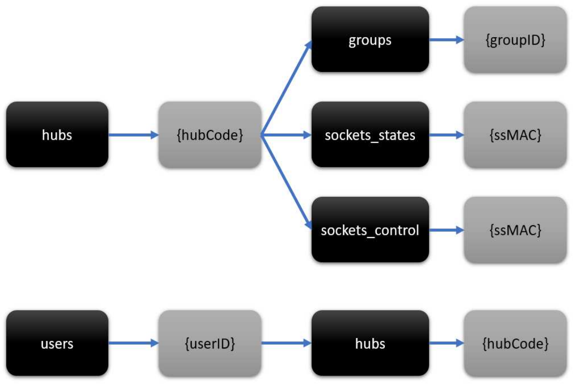
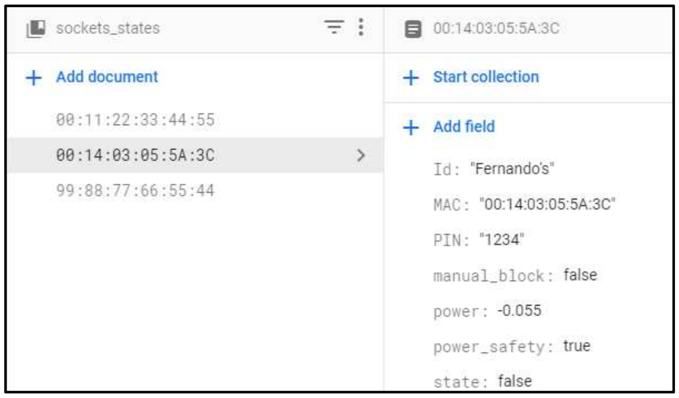


Figure 7. Firebase Firestore Database Configuration

## D. *SS Android App Implementation*

Regarding the mobile application, a customized Android app has been developed using Android Studio as the IDE and Java for Android as the programming language.



sockets_states

Add document

00:11:22:33:44:55

00:14:03:05:5A:3C

99:88:77:66:55:44

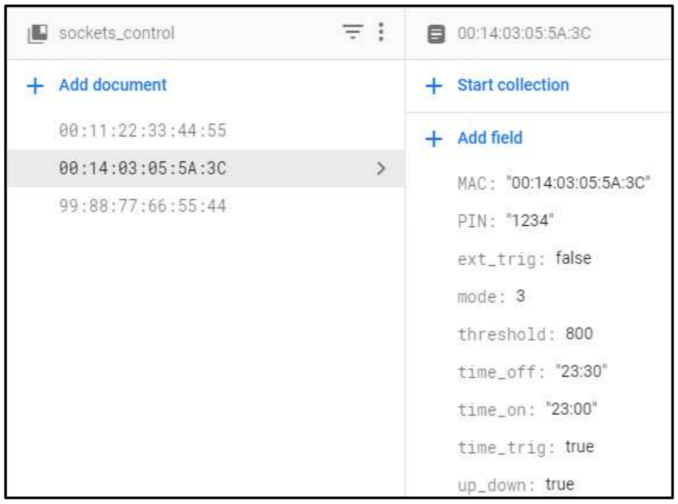
00:14:03:05:5A:3C

Start collection

Add field

```
Id: "Fernando's"
MAC: "00:14:03:05:5A:3C"
PIN: "1234"
manual_block: false
power: -0.055
power_safety: true
state: false
```

(a)



sockets_control

Add document

00:11:22:33:44:55

00:14:03:05:5A:3C

99:88:77:66:55:44

00:14:03:05:5A:3C

Start collection

Add field

```
MAC: "00:14:03:05:5A:3C"
PIN: "1234"
ext_trig: false
mode: 3
threshold: 800
time_off: "23:30"
time_on: "23:00"
time_trig: true
up_down: true
```

(b)

Figure 8. Example of Firebase Firestore Collections and Documents; (a) “sockets_control”, (b) “sockets_states”

The first task that the application needs to take care of is user management. For that, both registration and login activities are created. When a user is registered, a unique id is automatically assigned to him by the Firebase Authentication service, and the app is responsible for creating a document under the users' collection with that user ID.

Once that the user is logged in, the Main Activity of the customized Android app will launch where a list of all the SS Hubs and a button to add them are displayed. If the user presses the “add hub” button, a dialog fragment asking for the hub code and a hub name will appear.

Then, in the Hub Activity, the user will have two different tabs, one with a list of the SS and the other with a list of the groups. For adding the SS and creating groups, the user can press the corresponding “Add Smart Socket” or “Add Group”, which will show different Dialog Fragment. The SS one will ask for the MAC, PIN, and name, while the group one will ask for a group name and will display a list for selecting the desired groups.

Lastly, with respect to the SSs control, the user can press any SS in the list and that action will redirect the user to a Control SS Activity, in which the modes, time schedule, and the external trigger can be managed. Moreover, when there is any electrical issue in the circuit, an alert icon will appear, as well as a lock icon if the SS is manually blocked. Analog to this activity is the one for controlling the groups, in which the user has the same features, that when applied will be sent to all the SS in the group “sockets_control” collection. For a SS it exists the possibility of removing it from the hub, whereas for the group, the members can be modified or the group can be removed.

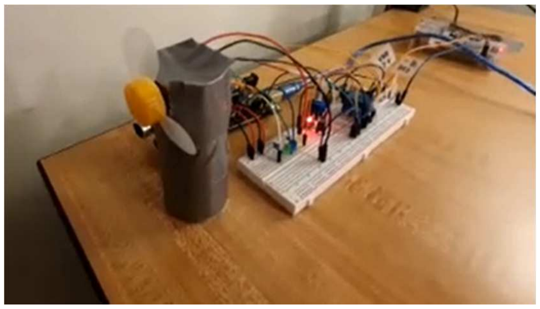
## IV. Results

In order to validate the prototyped SS operation capability, a 6 V electric fan has been used as a load, and as aforementioned SS hub has been used as a high voltage source. Furthermore, the local SS has been fed using the USB connection from a laptop. Using this test setup, all the following features have been tested:

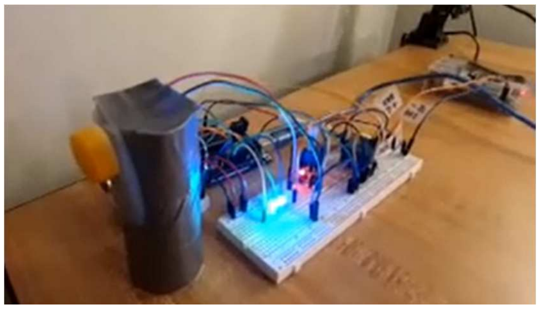
- Manual mode and automatic modes
- Scheduling
- Electrical safety
- SSs management and remote operation from app

The validation phase shows the following:

- The SS has been developed, reacting to the user's remote and local commands as shown in Figure 9.
- The SS successfully implements the electrical safety measurements, impeding any operation if the voltage and current levels are not within the acceptable range.
- The SS properly calculates the consumed power by the load.
- The SS properly establishes a Bluetooth-based connection with the SS hub and exchanges information successfully.
- The SS hub properly deals with information exchange with both the SSs and the remote server.



(a)



(b)

Figure 9. Prototype Validation Result; (a) SS Prototype ‘OFF’ State with All Other State LEDs Dimmed, (b) SS Prototype ‘ON’ Supplying Fan Load, Switch State and Blockage LED ‘ON’

Functioning Android app has been developed as shown in Figure 10 where the different screenshots of the Android app implementation show the following:

(a) User's SS tab Dashboard, with a list of all the available SS and another tab for the groups' list.
(b) Interface for new SS addition, asking for the PIN, the MAC address, and a name for the SS.
(c) Control activity for a whole group, including group management options in addition to the SSs control ones.
(d) Single SS control activity.
(e) Dialog with a checkbox for managing the SS selection when creating or modifying a group introduction.

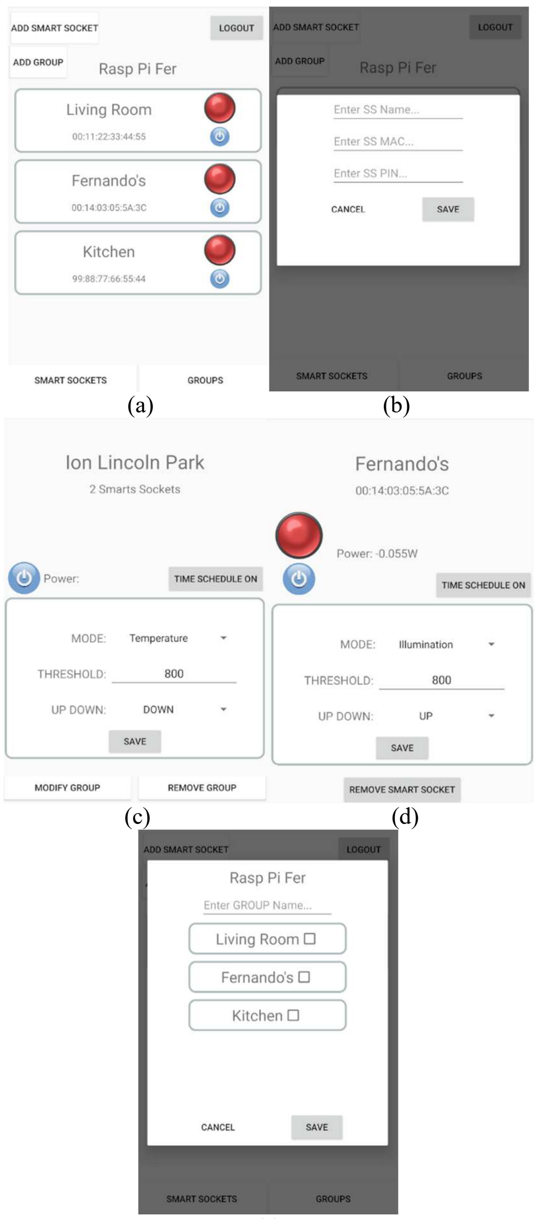


Figure 10. Android App Implementation; (a) User's SS Tab Dashboard, (b) Interface for Adding New SS, (c) Group-level Control, (d) SS-level Control, (e) SSs' Group Creation

## V. Conclusions

In this paper, a fully functional prototype of the smart socket system has been designed and realized with the ability of remote operations. A complete hardware design and implementation were carried out, in addition to the Arduino programming for controlling the system components. We utilized Python to realize the software design of the system. One of the most challenging parts of the implementation was achieving the scanning and pairing of the Bluetooth device using only a single script since it was needed to use subprocess routines for calling the *bluetoothcl* command and set a delay time for the scanning process. Also, cloud technologies were used via Firebase Authentication and Firestore database services. As future work, logging capabilities can be implemented in the SSs and, to add an advanced power usage optimization, machine learning algorithms can be applied to the data collected, predicting peak demand hours or off-peak hours, being able to, based on the users' usage patterns, enhance the energy consumption and reduce the electricity bill.

## References

[1] Leviton, "DW15P-1BW Decora Smart Wi-Fi Mini Plug-In", amazon.com. [Online]. Available: https://www.amazon.com/dp/B07H3JVZJZ?tag=techhivecom-20&th=1&psc=1&ascsubtag=US-003-3318250-001-1441895-web-20 [Accessed: Feb. 1, 2021].

[2] Currant, "Smart Plug WiFi Outlet with Energy Monitoring", amazon.com. [Online]. Available: https://www.amazon.com/dp/B07HKVRGNN?tag=techhivecom-20&th=1&psc=1&ascsubtag=US-003-3318241-005-1442076-web-20 [Accessed: Feb. 1, 2021].

[3] Lutron, "Caseta Smart Home Plug-In Lamp Dimmer", amazon.com. [Online]. Available: https://www.amazon.com/dp/B00JJY1QG0?tag=techhivecom-20&th=1&psc=1&ascsubtag=US-003-3322962-000-1441998-web-20 [Accessed: Feb. 1, 2021].

[4] iHome, "HomeKit ISP6X Wi-Fi Smart Plug", amazon.com. [Online]. Available: https://www.amazon.com/dp/B01HCVG9NG?tag=techhivecom-20&th=1&psc=1&ascsubtag=US-003-3185744-001-1437824-web-20 [Accessed: Feb. 1, 2021].

[5] Apromio, "Smart Plug Wi-Fi Enabled Mini Smart Socket", amazon.com. [Online]. Available: https://www.amazon.com/Enabled-Socket-Compatible-Control-Function/dp/B075GV6RLJ?th=1 [Accessed: Feb. 1, 2021].

[6] Aoycocr, "WiFi Smart Plug", amazon.com. [Online]. Available: https://www.amazon.com/Aoycocr-Energy-Wireless-Required-Compatible/dp/B07RXDHMCY [Accessed: Feb. 1, 2021].

[7] G. Song, F. Ding, W. Zhang and A. Song, "A wireless power outlet system for smart homes", *IEEE Trans. Consum. Electron.*, vol. 54, no. 4, pp. 1688-1691, Nov. 2008.

[8] J. Han, C.-S. Choi and I. Lee, "More efficient home energy management system based on ZigBee communication and infrared remote controls", *IEEE Trans. Consum. Electron.*, vol. 57, no. 1, pp. 85-89, Feb. 2011.

[9] A. H. Shajahan and A. Anand, "Data acquisition and control using arduino-Android platform: Smart plug", *Proc. Int. Conf. Energy Efficient Technol. Sustainability*, pp. 241-244, Apr. 2013.

[10] T. M. Fernández-Caramés, "An intelligent power outlet system for the smart home of the Internet of Things", *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 11, pp. 214805, 2015.

[11] M. S. Ahmed et al., "Smart plug prototype for monitoring electrical appliances in home energy management system", *Proc. IEEE Student Conf. Res. Develop.*, pp. 32-36, Dec. 2015.

[12] A. Singaravelan and M. Kowsalya, "Design and implementation of standby power saving smart socket with wireless sensor network", *Procedia Comput. Sci.*, vol. 92, pp. 305-310, Aug. 2016.

[13] V. Potdar, A. Sharif and E. Chang, "Wireless Sensor Networks: A Survey", *2009 International Conference on Advanced Information Networking and Applications Workshops, Bradford*, 2009, pp. 636-641.