How to make IoT in C# using Sinric Pro

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information Sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in
[2] Vice Chancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Area of the Paper: Computer Science.
Type of the Paper: Simulation-based Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: https://doi.org/10.5281/zenodo.7335167
Google Scholar Citation: IJCSBE

How to Cite this Paper:
Chakraborty, S., & Aithal, P. S., (2022). How to make IoT in C# using Sinric Pro. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *6*(2), 523-530. DOI: https://doi.org/10.5281/zenodo.7335167

International Journal of Case Studies in Business, IT and Education (IJCSBE)
A Refereed International Journal of Srinivas University, India.

Crossref DOI: https://doi.org/10.47992/IJCSBE.2581.6942.0215

Paper Submission: 28/10/2022
Paper Publication: 17/11/2022

Disclaimer: The scholarly papers as reviewed and published by the Srinivas Publications (S.P.), India are the views and opinions of their respective authors and are not the views or opinions of the S.P. The S.P. disclaims of any harm or loss caused due to the published content to any party.

How to make IoT in C# using Sinric Pro

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information Sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] ViceChancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *Nowadays, IoT, or the internet of things, is a popular technological term. Any device operation is possible anywhere in the world using IoT. It controls the Device using a couple of telemetry protocols. Here we will describe how we can experiment with IoT using the Sinric Pro IoT platform. We will test without any hardware. We will also operate our Device from Alexa and Sinric Pro mobile app. the IoT client is written in C#. The complete project code is adopted from Sinric Pro. The Researcher can modify and integrate the code as their research needs.*

Design/Methodology/Approach: *Sinric Pro is a popular IoT service. First, we create our account on the Sinric Pro website. Then we configure devices and note the API key, secret key, and device id. We add those keys to our C# application and run the client application. The running client receives a command from various sources. We also Configure the mobile application to send the command through apps.*

Findings/Result: *This procedure helps us understand the data flow between the IoT server and the client. We can understand what happens inside the client application using a state diagram. We can use it as a demonstration tool and possibly to trigger the actual Device. Fetching the load status from the client application might trigger the controller board through the serial port.*

Originality/Value: *Sometimes Researcher cannot get the actual hardware for the experiment due to availability or cost constraints. This procedure might help them. Working with existing hardware also need some basic electronics knowledge. Otherwise, severe damage can occur inside the working system. There is no such risk in this procedure, and it produces the result efficiently.*

Paper Type: *Simulation-based Research.*

Keywords: IoT Demonstration. Sinric Pro, Sinric Pro Setup.

1. INTRODUCTION :

We found the term IoT for more than a decade. It is becoming popular day by day. Lots of boards are also available to work efficiently with IoT. Round the clock, the Researcher is engaged in providing more IoT-enabled devices. Here we will do experiments with IoT devices. In C#, we will create IoT client applications that act as physical devices. It is transparent to the Server. The IoT server will not be aware the Device or endpoint is running in a virtual platform. The request and response and housekeeping functions as the same as real hardware. Sinric Pro provides the SDK. Using that, we customized it to our needs.

2. RELATED WORKS :

Datta, S. K. et al. proposed an edge-computing architecture for IoT data through an intermediate computing layer. They use virtual IoT devices for local data processing and take quick action to the actuator. The application of their system is for roadside assistance and autonomous vehicle [1]. In their paper, Datta, S. K., & Bonnet, C. proposed a procedure to create and operate a virtual IoT device that can be a virtual sensor and an actuator [2]. Lee, B. M. et al. proposed an intelligent service model for the healthcare industry which provides adequate feedback to an individual entity [3]. Hayashi, V. T. et al. provide a scheme-based protocol with IoT autonomous devices without additional user interaction. They used a web socket with an average response time [4]. Alabady, S. A., et al. present and discuss a typical network security model for cooperative virtual networks in the IoT era. Their paper provides the fundamentals of a secure networking system, including a firewall. It presents a novel security model to defend the network from internal and external attacks and threats in the IoT Era [5].

3. OBJECTIVES :

Our objective is to provide reference information to the Researcher working to integrate IoT into their projects. Here we provide the complete details on how to create an IoT account, configure the Device, and trigger it using the Sinric Pro platform.

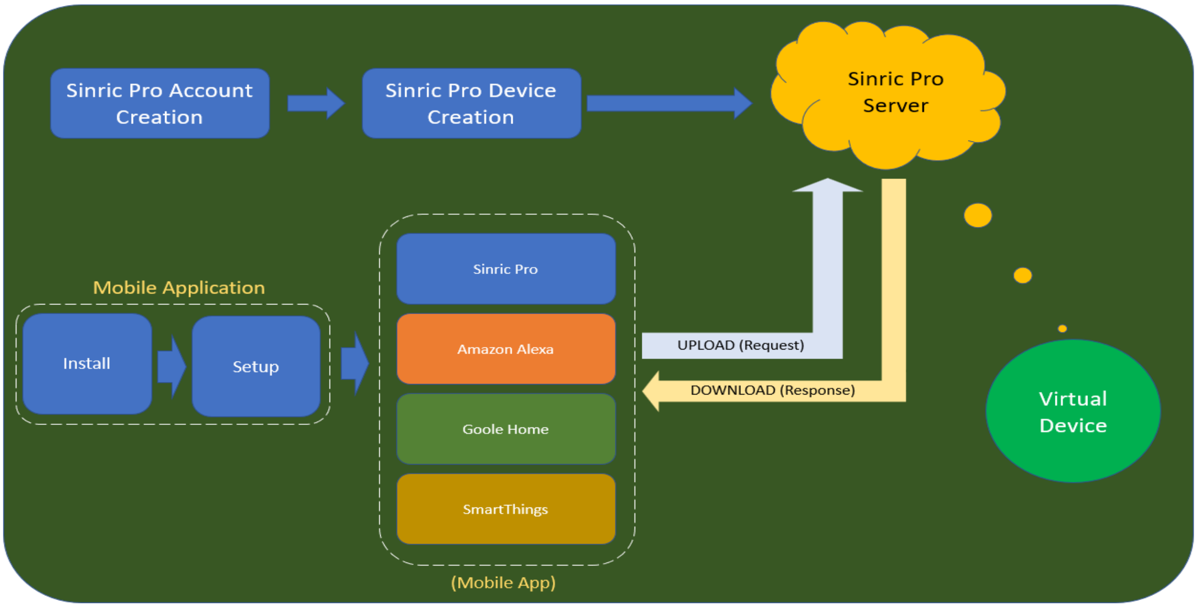


Fig.1: Block diagram of the steps to work

4. APPROACH AND METHODOLOGY :

Figure 1 depicts the steps for our work. We create an account inside the Sinric Pro IoT server. After, we configure devices in the Server. Then download the IoT C# client. We configure the IoT client for created Device. We installed the Mobile application and configured it. We trigger the load from anywhere, and the Device gets on/off on the command.

Sinric Pro account creation: This is part of the server-side setup. We need to open the web page inside a browser. Go to https://sinric.pro/index.html. Click on "Sign up!". The page will appear as depicted in figure 2. fill up the form is self-explanatory. Last, click on "I'm not a robot" and then click the "Register" button. Sinric Pro will send a confirmation mail to the registered email ID. Open the mail and confirm. Now login process into the Sinric Pro web server is completed.

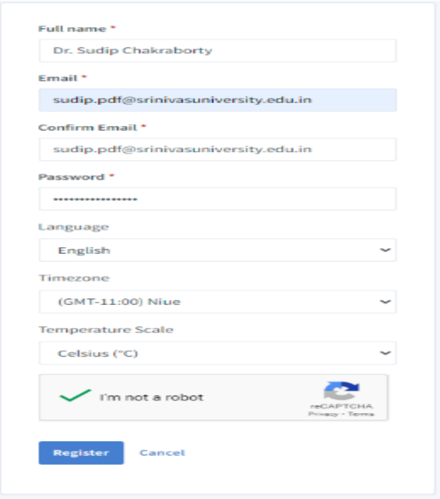

Full name *
Dr. Sudip Chakraborty
Email *
sudip.pdf@srinivasuniversity.edu.in
Confirm Email *
sudip.pdf@srinivasuniversity.edu.in
Password *
•••••••••••••••••
Language
English
Timezone
(GMT-11:00) Niue
Temperature Scale
Celsius (°C)
✓ I'm not a robot

Fig.2: Signup Page

Device Creation:

This is another basic server-side setup. We will create the Device on the Server. On the left side of the page is a "**Device**" button depicted in figure 4.3. Click on it. There is a "**+Add Device**" button on the right side pan. Click on it. The new device page will appear. Fill in The Device name "**DemoLock**." In the description, add "**Demonstration Lock**." Under the Device Type from the list, select “**Smart Lock**.” press the “**Next**” Key. To view every activity, enable all alerts by changing the slider position. Press “**Next**,” again press “**Next**.” Then press the “**Save**” button. Now our Device is ready to connect. The Three generated IDs need to copy. The Device id, App Key, and App Secret key are depicted in figure 4. These keys are required for the client application.

IoT Client Preparation in C#:

We follow the below steps to set up a C# client.

1) Install the visual studio community edition. When we install, we have to select C# related component to install.
2) Open Sinric pro wesite home page. Under SDK, click on “*C#(Community Maintained*). It will navigate to the GitHub repository download page. Go to code and download the zip file.
3) Unzip the downloaded file sinric-pro-CSharp-master—double-click on “**main.sln”.** The file will open in visual studio.
4) The right side is the Solution explorer window. Under “ConsoleExampleCore,” delete the appsettings.private.json file.
5) The “Build Solution” is available under the “**Build**” menu. Build the solution. It should build the entire solution successfully.
6) From the right side, Open the “**appsettings.json”** file. Modify it according to the server configuration depicted in figure 5.
7) After building the project, when run, some error related to the "Kitchen Door" may arise. We can suppress the error by commenting on the line depicted in figure 6,

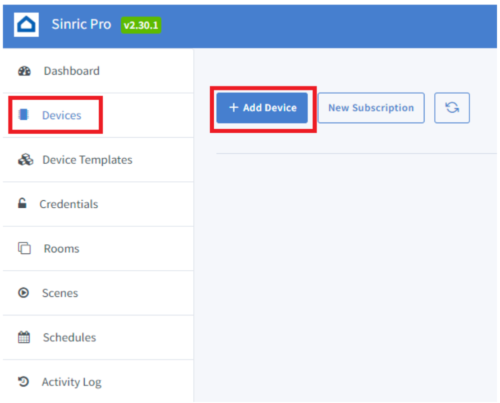
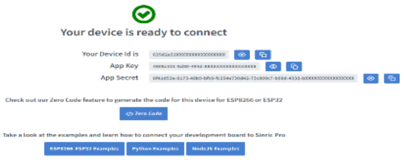


Fig.3: Device addition page



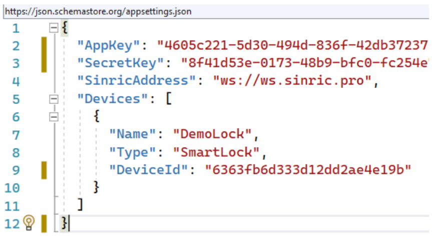
Your device is ready to connect

Your Device Id is 635d2a330000000000000000
App Key 4605c221-5d30-494d-XXXXXXXXXXXXXXXXXX
App Secret 8f41d53e-0173-48b9-bfc0-fc254e730d42-73c900c7-b88d-4330-b00000000000000000

Check out our Zero Code feature to generate the code for this device for ESP8266 or ESP32

Take a look at the examples and learn how to connect your development board to Sinric Pro

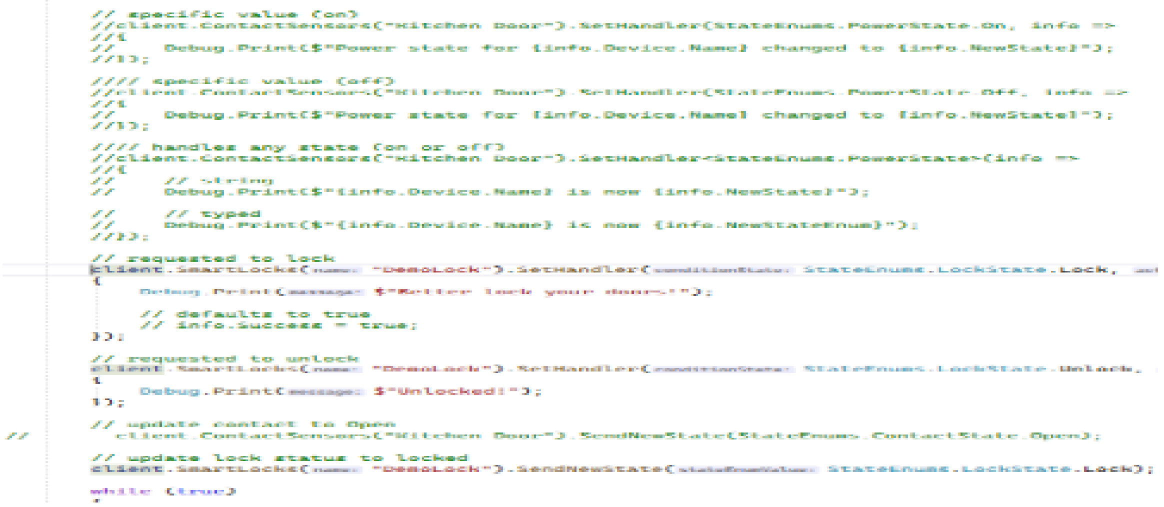
Fig. 4: Key View



https://json.schemastore.org/appsettings.json

```json
{
  "AppKey": "4605c221-5d30-494d-836f-42db37237
  "SecretKey": "8f41d53e-0173-48b9-bfc0-fc254e
  "SinricAddress": "ws://ws.sinric.pro",
  "Devices": [
    {
      "Name": "DemoLock",
      "Type": "SmartLock",
      "DeviceId": "6363fb6d333d12dd2ae4e19b"
    }
  ]
}
```

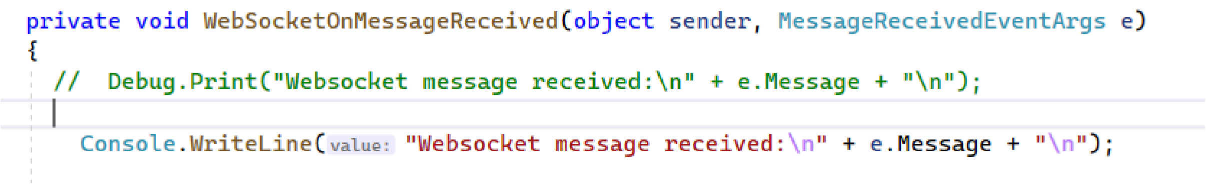
Fig. 5: Update Json File



```csharp
      // specific value (on)
      //client.ContactSensors("Kitchen Door").SetHandler(StateEnums.PowerState.On, info =>
      //{
      //    Debug.Print($"Power state for {info.Device.Name} changed to {info.NewState}");
      //});

      //// specific value (off)
      //client.ContactSensors("Kitchen Door").SetHandler(StateEnums.PowerState.Off, info =>
      //{
      //    Debug.Print($"Power state for {info.Device.Name} changed to {info.NewState}");
      //});

      //// handles any state (on or off)
      //client.ContactSensors("Kitchen Door").SetHandler<StateEnums.PowerState>(info =>
      //{
      //    // string
      //    Debug.Print($"{info.Device.Name} is now {info.NewState}");

      //    // typed
      //    Debug.Print($"{info.Device.Name} is now {info.NewStateEnum}");
      //});

      // requested to lock
      client.SmartLocks(name: "DemoLock").SetHandler(conditionState: StateEnums.LockState.Lock, as
      {
          Debug.Print(message: $"Better lock your door~!");

          // defaults to true
          // info.Success = true;
      });

      // requested to unlock
      client.SmartLocks(name: "DemoLock").SetHandler(conditionState: StateEnums.LockState.Unlock,
      {
          Debug.Print(message: $"Unlocked!");
      });

      // update contact to Open
//      client.ContactSensors("Kitchen Door").SendNewState(StateEnums.ContactState.Open);

      // update lock status to locked
      client.SmartLocks(name: "DemoLock").SendNewState(stateEnumValue: StateEnums.LockState.Lock);

      while (true)
      {
```

Fig.6: Suppress the error related to “Kitchen Door”

8) One another small change needs to be done inside SinricClient.cs module around 182 lines, like in figure 7.



```
private void WebSocketOnMessageReceived(object sender, MessageReceivedEventArgs e)
{
  //  Debug.Print("Websocket message received:\n" + e.Message + "\n");
  |
    Console.WriteLine(value: "Websocket message received:\n" + e.Message + "\n");
```

Fig.7: Change to display the receive packet inside the console window

Now, our IoT client is ready to test. Build and Run the application.

9) Open the Dashboard of the Sinric Pro website. Press the "**Lock**" and "**Unlock**" buttons. Observe the console windows displaying messages upon receiving the packet when we press the dashboard button, depicted in figure 8, we triggered from Cloud Dashboard. We can also operate from Alexa mobile apps using Voice commands.

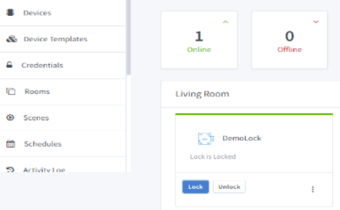
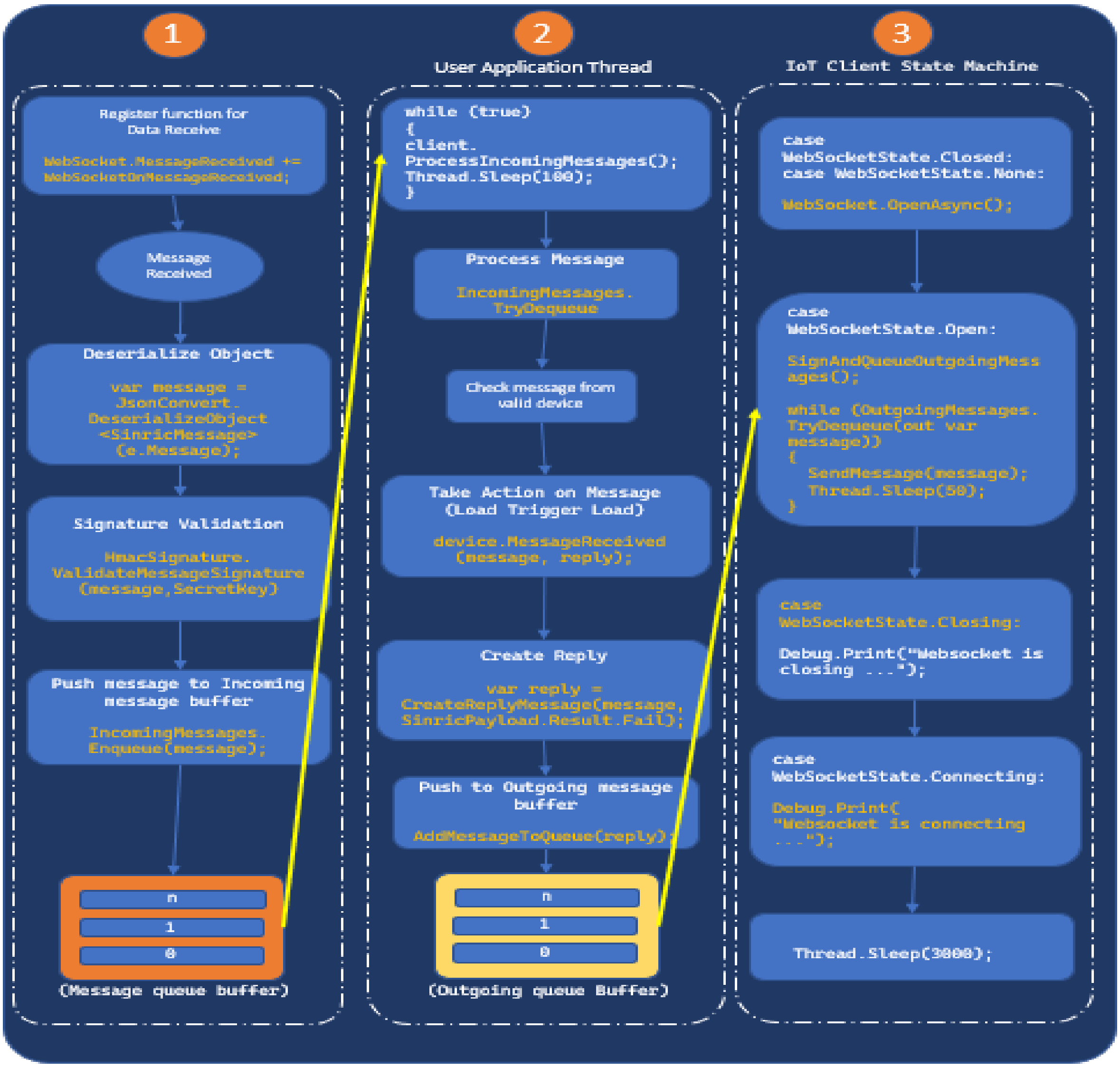


Fig. 8: Sinric Pro Dashboard



```
WebSocket.MessageReceived +=
WebSocketOnMessageReceived;
```

```
var message =
JsonConvert.
DeserializeObject
<SinricMessage>
(e.Message);
```

```
HmacSignature.
ValidateMessageSignature
(message,SecretKey)
```

```
IncomingMessages.
Enqueue(message);
```

```
while (true)
{
client.
ProcessIncomingMessages();
Thread.Sleep(100);
}
```

```
IncomingMessages.
TryDequeue
```

Check message from valid device
Take Action on Message (Load Trigger Load)

```
device.MessageReceived
(message, reply);
```

```
var reply =
CreateReplyMessage(message,
SinricPayload.Result.Fail);
```

```
AddMessageToQueue(reply);
```

```
case
WebSocketState.Closed:
case WebSocketState.None:

WebSocket.OpenAsync();
```

```
case
WebSocketState.Open:

SignAndQueueOutgoingMess
ages();

while (OutgoingMessages.
TryDequeue(out var
message))
{
   SendMessage(message);
   Thread.Sleep(50);
}
```

```
case
WebSocketState.Closing:

Debug.Print("Websocket is
closing ...");
```

```
case
WebSocketState.Connecting:

Debug.Print(
"Websocket is connecting
...");
```

```
Thread.Sleep(3000);
```

Fig. 9: C# IoT Client Finite State Diagram

Figure 9 depicts the process flow of the C# client application. In the first step, we register one receive function for data reception. When data reveice, the data flows through the process and push to the message queue buffer. Step two processes the data, takes action, and prepares the reply message. Then it sends it to the outgoing message buffer. In the third step, the packet is sent to the IoT server.

5. EXPERIMENT :

Now, we will experiment with the Sinric pro mobile application.

Sinric Pro Application setup: Sinric Pro has native apps to interact with the Server. To do that, search Sinric Pro and install it in the play store. After installation, log in using the credential. After login, we will see the Device we added to the Sinric Pro website. Run C# application. We will get a notification in the apps the Device is online. Now from mobile apps, we can operate our Devices; here, we can lock and unlock our Devices through mobile.

Amazon Alexa setup (Android Mobile): The following step can help us to sync between sinric Pro and Alexa.
1) Install Alexa App
2) Open the App. We must sign in using an Amazon account using an Email or phone number and an Amazon password. If an account does not exist, we need to create one.
3) At the bottom, press the "**Devices**" button tab. It will open a new page.
4) At the top right, press the All Devices (Three Dots) Button.
5) At the top right, press the "+" Button.
6) Press on "**Add Devices**."
7) On the "Device type or brand" textbox, type "**Sinric Pro**." Click on the link "**Go to the skill store**." In the search result, at the top, find "**Sinric Pro**" and press on it.
8) Click the "**ENABLE TO USE**" Button
9) Enter Sinric Pro Login Credential, i.e., email id and password. Check" I'm not a robot." and press "Sign in.". press "**next**."
10) It will take a while. One message will display, "**Alexa is looking for devices to connect….this may take up to 45 seconds**". After a while. It will connect with Sinric Pro IoT Server. Close the Alexa. The Sinric Pro account is synced With Amazon Alexa Apps. Close the Alexa app.

Operate Device from the Alexa app: Now, we will see how we can enable voice commands on our devices.
1. Open the Alexa app.
2. At the bottom, press the "**Devices**" Button.
3. At the top, we will see our devices. Press on the "**Lock**" icon. Then you will see "**DemoLock**." Press on it. We may see "**DemoLock is Locked. Unlocking is disabled by default. To enable unlocking by app, go**…."
4. At the top, press the settings button. Change the slide for "**unlock by app**" For this process, it might be asked for an amazon password.
5. For "**unlock by voice**." The app will be asked to enter a pin. The lock is a sensitive Device. To protect from unlocking, only anyone's voice needs extra protection. Provide a PIN and enter again for confirmation. Press the "**Back**" Button.
6. Open the C# application. And RUN it.
7. Now Press the "**Lock**" button and observe the C# terminal message. And inside the app, we will simultaneously see the lock, unlock and change status messages.

Alexa voice command setup for lock:
1. Open Alexa. Go to devices. At the top, click the "Lock" icon.
2. Click on "**DemoLock**." At the bottom. Press "**Create Routine**."
3. Click "**Enter routine name**" and click on "+" Sign.
4. Here we enter the" **device lock routine**." At the top right, press next. It is just the Name of a routine or function.

5. Now, this is the vital part. Press on "**When this happens**." Next, on "**Voice**." type what we will say inside the textbox, here we say, "**please lock**." “Alexa” by default prefix. No need to add “**Alexa**.” press next.
6. Now press on "Add action." click on "Smart Home" > DemoLock>press Add.
7. Click save. Our routine is ready.
8. Now say, “**Alexa, please lock**.” We will see client received the message. And mobile apps also got the status change message.
9. Using the above method, we can create different routines for different actions.

6. RECOMMENDATIONS :

a) For Microsoft visual studio community edition - https://visualstudio.microsoft.com/vs/community/
b) For Amazon Alexa setup, one reasonable effort video: https://www.youtube.com/watch?v=zXdBcfJYG5E
c) For Github C# code for Sinric Pro https://github.com/xamakadesigns/sinric-pro-csharp
d) The experimental code is available: at https://github.com/sudipchakraborty/How-to-make-IoT-In-C-using-Sinric-Pro.git

7. CONCLUSION :

Through this research work, we see how we can set up an experiment with IoT. We select the popular IoT server Sinric pro. Using this platform, we operate our Devices from various IoT mobile applications. Through these documents, the Researcher can create their IoT-enabled Device. This procedure can be used for Demonstration or to develop some work without requiring the actual hardware.

REFERENCES :

[1] Datta, S. K., & Bonnet, C. (2017, October). An edge computing architecture integrates virtual IoT devices. In 2017 IEEE 6th Global Conference on Consumer Electronics (GCCE) (pp. 1-3). IEEE. Google Scholar↗

[2] Datta, S. K., & Bonnet, C. (2017, June). Extending data tweet IoT architecture for virtual IoT devices. In 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSC), and IEEE Smart Data (SmartData) (pp. 689-694). IEEE. Google Scholar↗

[3] Lee, B. M., & Ouyang, J. (2014). Intelligent healthcare service by using collaborations between IoT personal health devices. International Journal of Bio-Science and Bio-Technology, 6(1), 155-164. Google Scholar↗

[4] Hayashi, V. T., & Ruggiero, W. V. (2022). Hands-Free Authentication for Virtual Assistants with Trusted IoT Devices and Machine Learning. Sensors, 22(4), 1325. Google Scholar↗

[5] Alabady, S. A., Al-Turjman, F., & Din, S. (2020). A novel security model for cooperative virtual networks in the IoT era. *International Journal of Parallel Programming*, *48*(2), 280-295. Google Scholar↗

[6] Sahlmann, K., & Schwotzer, T. (2018, October). Ontology-based virtual IoT devices for edge computing. In Proceedings of the 8th International Conference on the Internet of Things (pp. 1-7). Google Scholar↗

[7] Petrenko, A. S., Petrenko, S. A., Makoveichuk, K. A., & Chetyrbok, P. V. (2018, January). The IIoT/IoT device control model is based on narrow-band IoT (NB-IoT). The 2018 IEEE conference of young Russian researchers in electrical and electronic engineering (EIConRus) (pp. 950-953). IEEE. Google Scholar↗

[8] Celesti, A., Mulfari, D., Fazio, M., Villari, M., & Puliafito, A. (2016, May). Exploring container virtualization in IoT clouds. In 2016 IEEE international conference on Smart Computing (SMARTCOMP) (pp. 1-6). IEEE. Google Scholar↗

[9] Sindhwani, N., Maurya, V. P., Patel, A., Yadav, R. K., Krishna, S., & Anand, R. (2022). Implementation of intelligent plantation system using virtual IoT. The Internet of Things and its Applications (pp. 305-322). Springer, Cham. Google Scholar↗

[10] Simiscuka, A. A., Markande, T. M., & Muntean, G. M. (2019). Real-virtual world device synchronization in a cloud-enabled social virtual reality IoT network. IEEE Access, 7, 106588-106599. Google Scholar↗

****