CONTEST 2022

SMART HOME AUTOMATION USING W5100S-EVB-Pico

By Albin Joseph

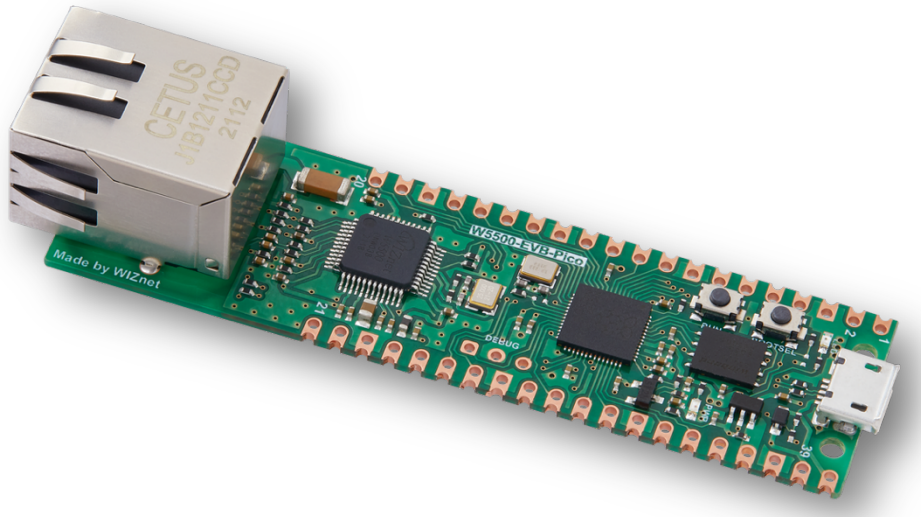
Contents:-

1) Quick Intro to W5100S-EVB-Pico
2) Intro to Project / Aim of the Project
3) Block Diagram of the Project
4) Hardware and Material Used
5) Problems Challenges and Solutions
6) R&D
7) Flow Diagram for Alexa and Google Assistant with Developer Console Images
8) Sample Shortcuts Code and Shortcuts Achievement
9) Circuit Python Code and Adafruit IO Dashboard
10) Results and Conclusion

Problem Statement

To convert normal home devices to smart home devices by creating an automated environment at a cheaper cost by making use of W5100s-evb-pico with internet connectivity and by proving it with a mini home demo model to create a similar environment as that of an actual home.

1) Quick Intro to W5100S-EVB-Pico



W5100S-EVB-Pico is a microcontroller evaluation board based on the Raspberry Pi RP2040 and fully hardwired TCP/IP controller W5100S – and basically works the same as Raspberry Pi Pico board but with additional Ethernet via W5100S.

- Raspberry Pi Pico Clone
- Ethernet (W5100S Hardwired TCP/IP CHIP)

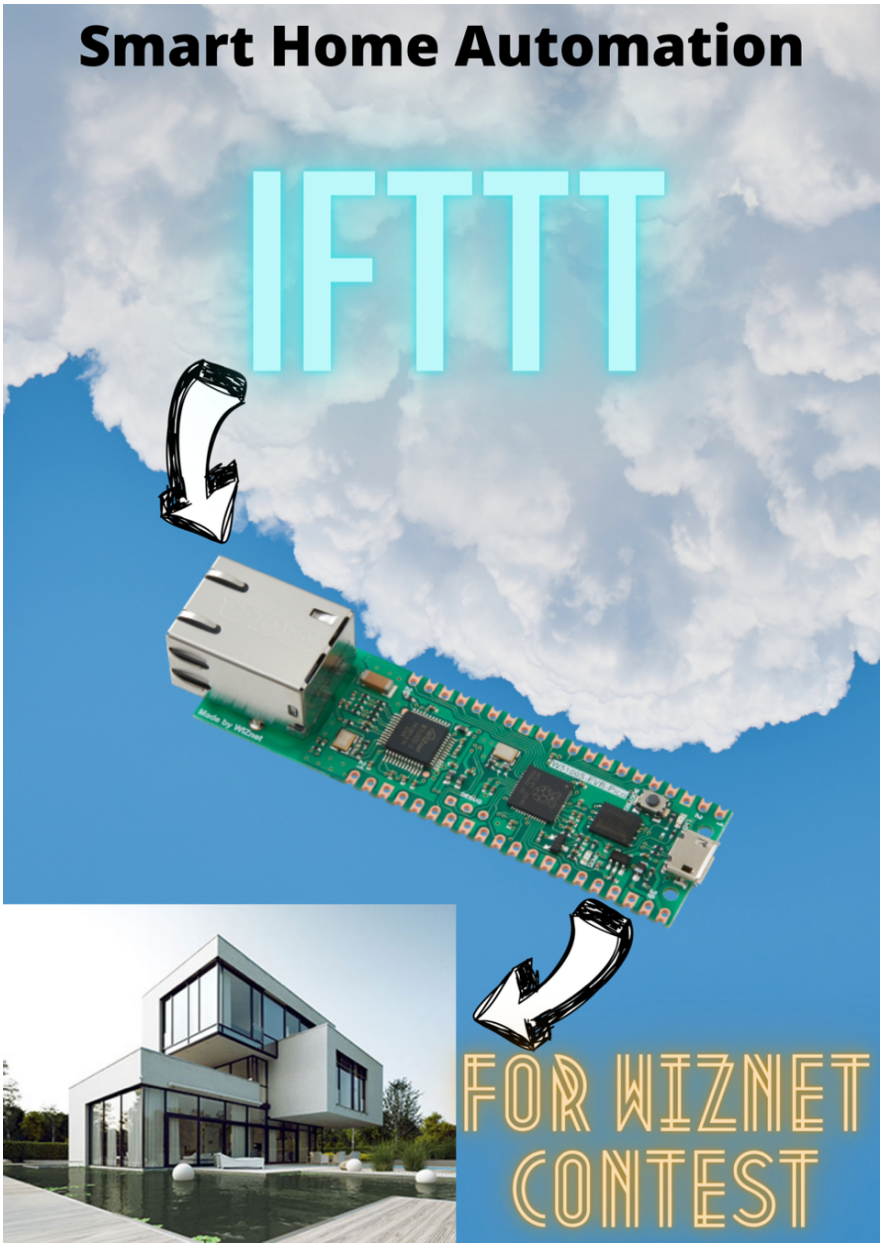
It is one of the best things on the planet from Wiznet!

More details can be found I the following link :- https://docs.wiznet.io/Product/iEthernet/W5100S/W5100S-evb-pico/

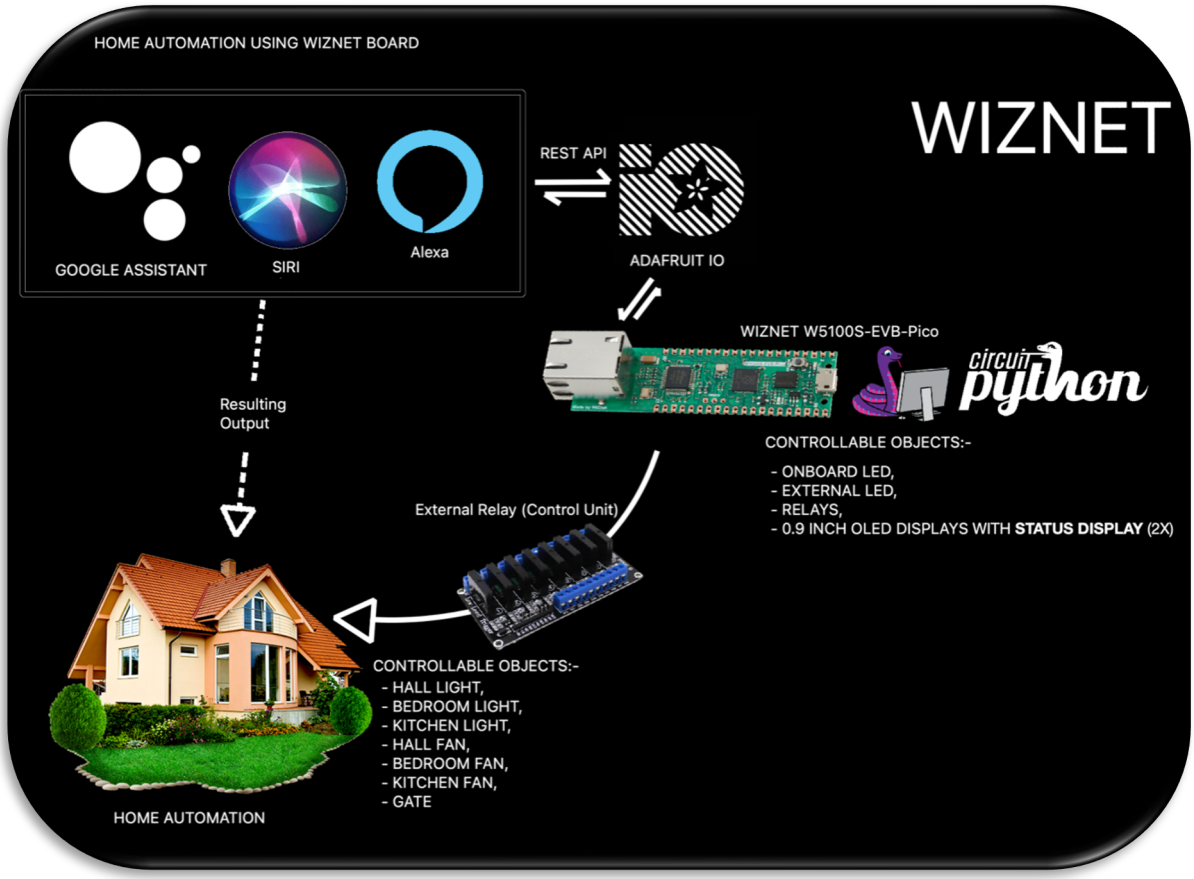
2) Intro to Project / Aim of the Project

The Aim of the project is to turn normal home devices to smart devices by creating an automated environment with the help of W5100s-evb-pico with internet connectivity to create a low cost home automation setup.

The Below was the proposed project flyer…. But things changed… changed for good… you will find out.



3) Block Diagram of the Project



The above Block Diagram shows the interaction flow (behind the scenes) of what actually happens when you interact with your favourite assistant !

4) Hardware and Material Used

W5100S-EVB-Pico ---------- x 1

0.9 Inch OLED Display ---------- x2

Bread Board ---------- x1

8 Channel Relay Module ---------- x1

Mini fans ---------- x2

DC Motor and Fan Blade ---------- x1

0.5µF capacitor for DC Motor ---------- x1

Acrylic Sheets 3 types ---------- x5

Plywood 2 types ---------- x2

False Ceiling Lights ---------- x3

Old CD/DVD Drive ---------- x1

5) Problems, Challenges and Solutions

There were many challenges faced starting from gathering the right parts but the main issue that was face was when the I tried to use a DC Motor fan, somehow the back EMF and magnetic interference was causing the relay module to stop functioning as soon as the motor starts and because the relay malfunctioned it was resetting the circuit python program and used to go on this reset loop finally causing the board to freeze and I had to end up nuking the board and re-flashing circuit python and the code.
Well actually there seemed two ways to solve this problem and the first and easy way was to replace the dc motor (which is used as hall fan) to a tiny propeller which would instantly solve the problem, but instead I went the tough way to solve this problem and started to learn about the reason behind it and solve it, it took me almost 3 weeks to find the solution to this, and at the end I was able to solve it using a 0.5µF capacitor which was connected in parallel to the DC motor along with a 1 ohm resistor, which did the job, but also a varistor could have been used too to solve this which unfortunately seemed to be very rare to find.

The other problem faced was to stick the Acrylic sheets together the available solvents and glue was not at all strong enough to hold it in place so I had to research and develop a solvent along with my father, the solvent was made by dropping acrylic pieces into Chloroform and using that solution to weld it together after cleaning up with alcohol The Acrylic sticks together due to chemical bonding.

6) R&D

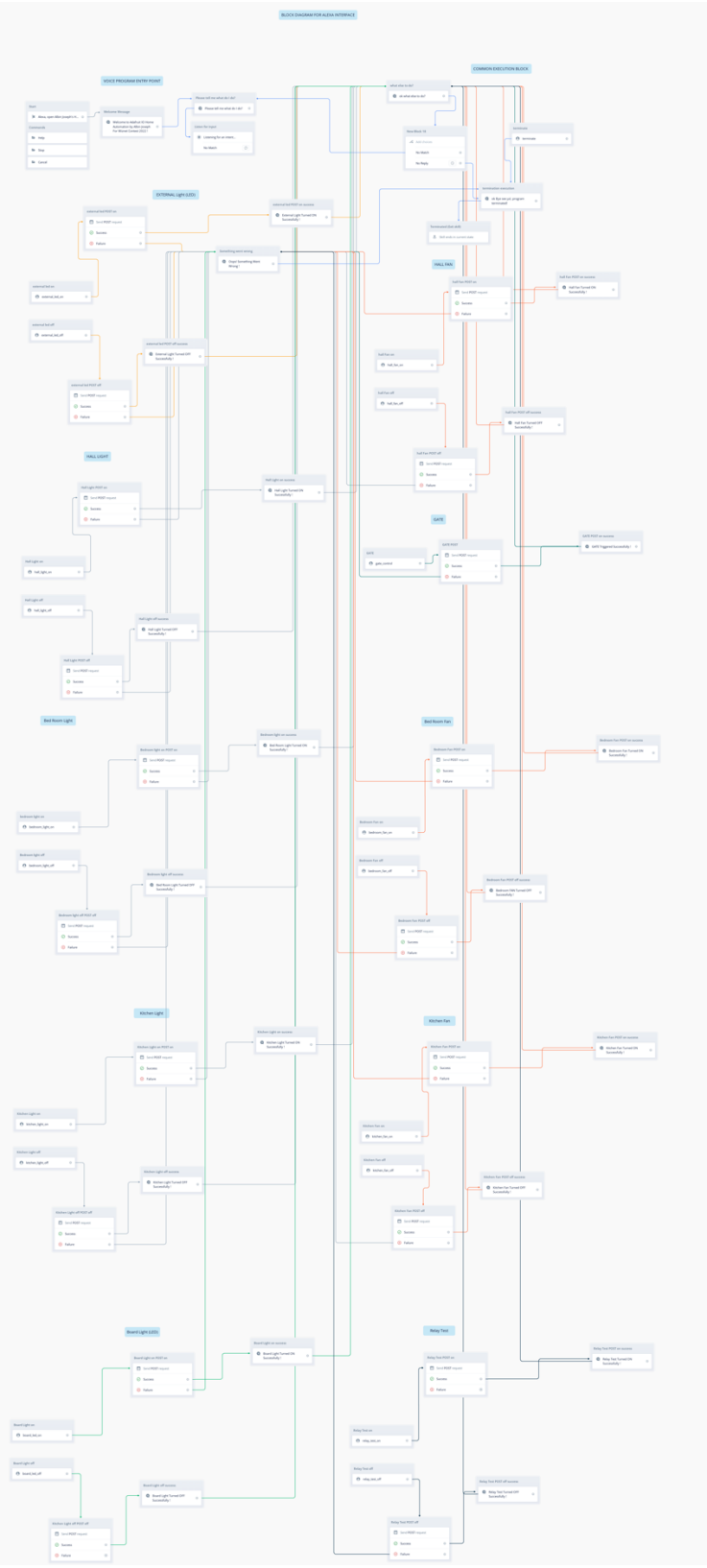
First thing to start off with the project was to create an actual house model and it was almost impossible to join the acrylic sheets together it seemed to fall off one by one causing a tiny domino effect. Somehow I had to figure out a way to fix those sheets together so after a bit of research me along with my dad was able to drop some acrylic sheets into chloroform to prepare its on solvent which made it bond at a molecular level.

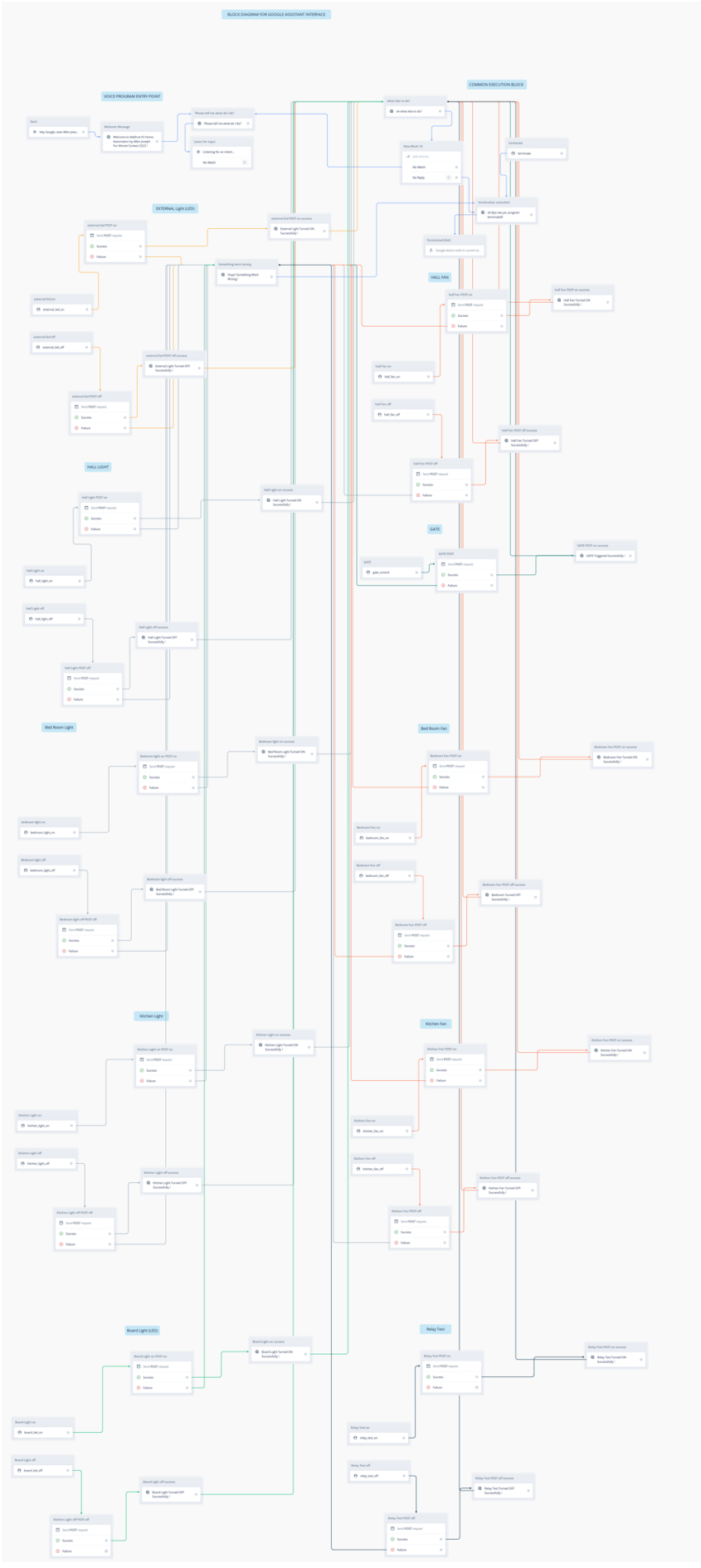
The other part where the R&D was done was on the software side… well I could have made use of IFTTT (Short for IF This Then That) a 3rd party service that can be used to connect Adafruit IO's feeds with Google Assistant or Alexa and can also be used to connect Siri with Adafruit IO's feeds by making use of webhooks provided by IFTTT which seemed easy and pretty straight forward right? And indeed it was my first plan and that is why you did see IFTTT on my initial project proposal flyer.

I was also wondering if I could just make Api calls and directly interact with Adafruit IO feeds as I was graving to know how it worked and I started to feel it to myself, "What if I could get rid of IFTTT, after all it is just a 3rd party" and so I ended up in this Adafruit IO's Api page which is located at https://io.adafruit.com/api/docs/#adafruit-io-http-api but most of it just showed how to fetch the data from in and not how to make use of it to trigger a feed, so after a few

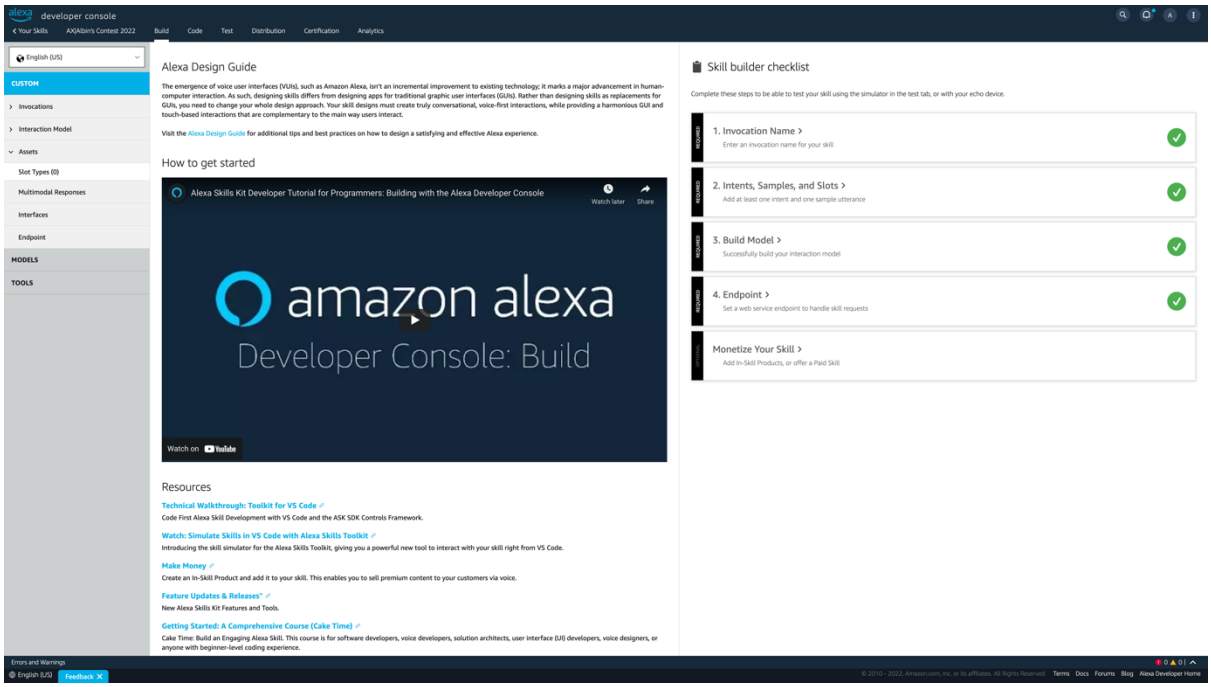
trial and error I was able to find the right method to send the value to trigger the feed using curl in terminal which led a way to completely let go of IFTTT and send out POST request to actually trigger the Adafruit Io's feed, it has also made it possible for me to later found a way to integrate it into Siri without using webhooks and directly sending requests to Adafruit IO's feeds and I can bet you can never find how it is done on shortcuts over the internet as I did search for the same across many forms, articles, YouTube videos and tutorial points they all just seemed know how to connect shortcuts to Adafruit IO's feed only with IFTTT. more of these details can be found in module 10 of this document.

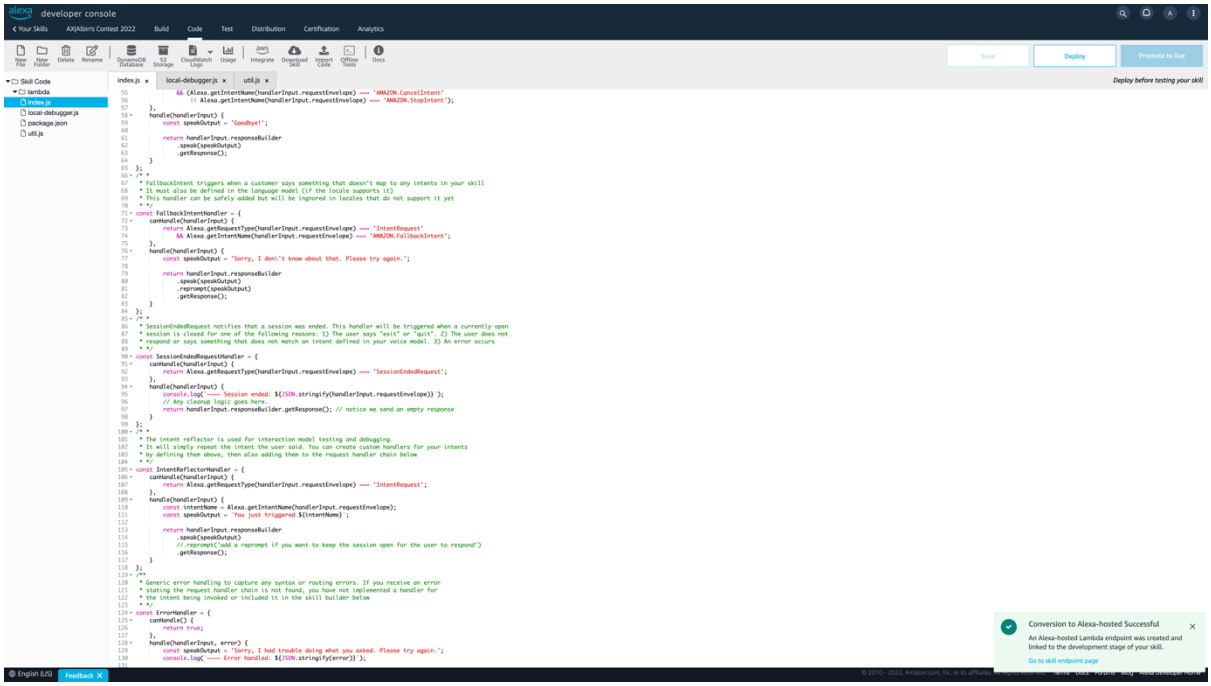
7) Flow Diagram for Alexa and Google Assistant



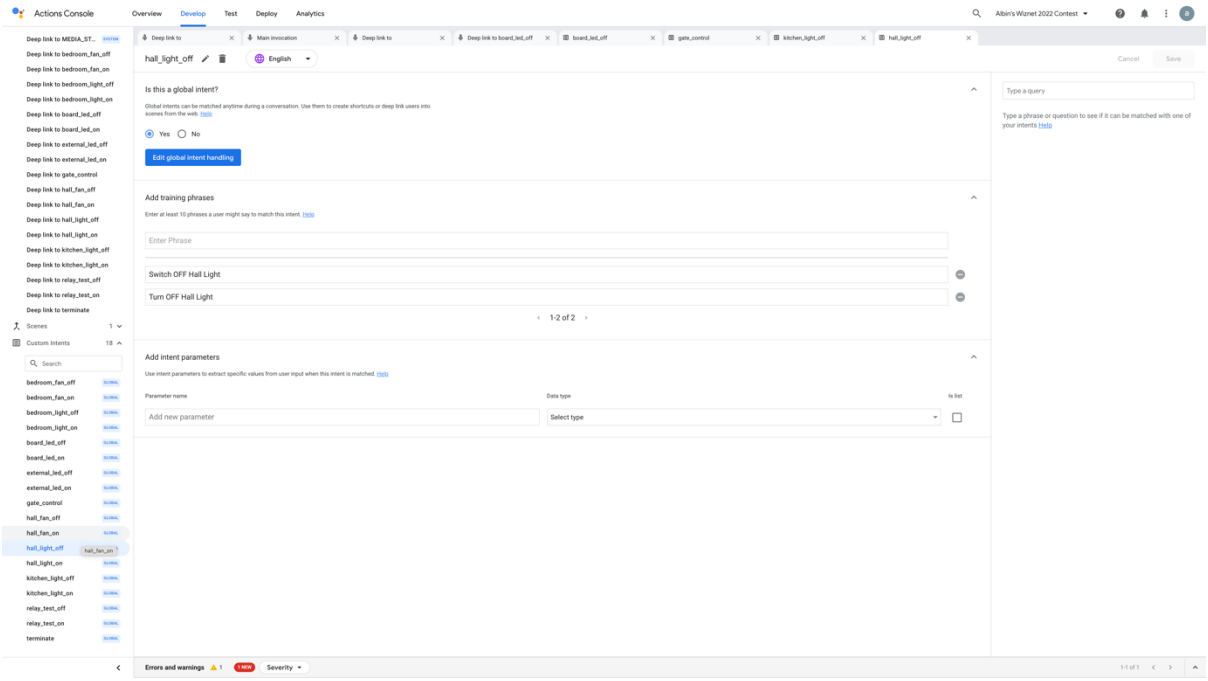

BLOCK DIAGRAM FOR GOOGLE ASSISTANT INTERFACE
VOICE PROGRAM ENTRY POINT
COMMON EXECUTION BLOCK
EXTERNAL Light (LED)
HALL FAN
HALL LIGHT
GATE
Bed Room Light
Bed Room Fan
Kitchen Light
Kitchen Fan
Board Light (LED)
Relay Test

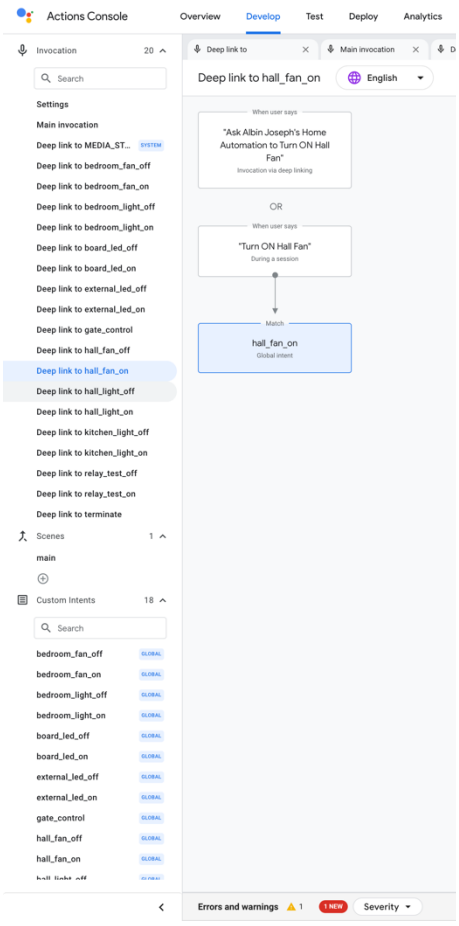
Alexa Developer console :-





Google Assistant Developer Console:-

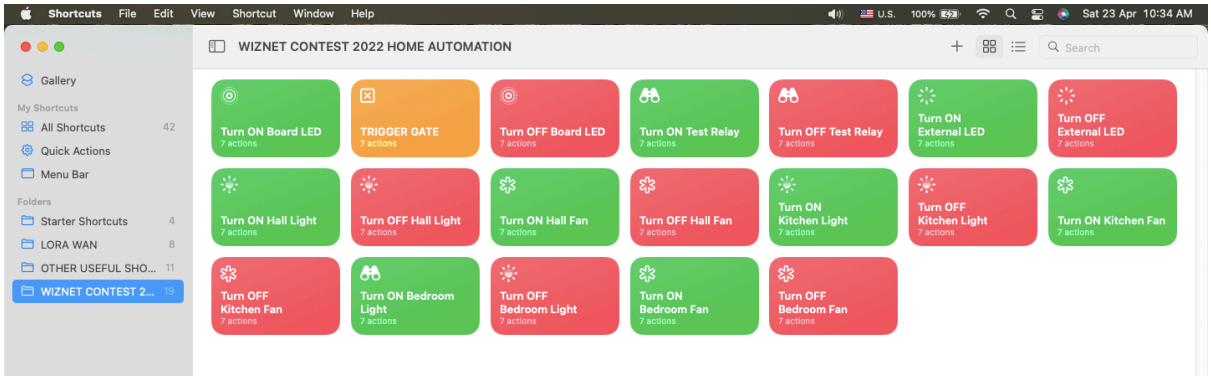




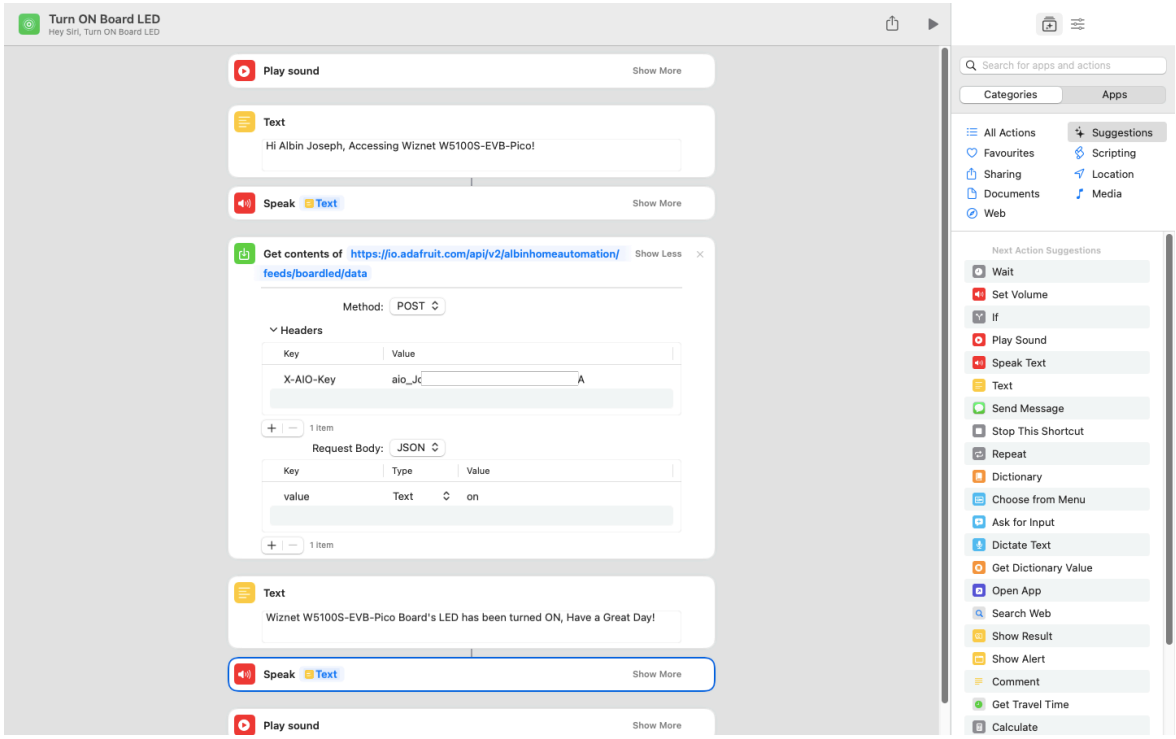
8) Sample Shortcuts Code and Shortcuts Achievement

The below pictures shows the shortcuts dashboard and a few code snippets:-

Siri Shortcuts Dashboard:-



Sample Shortcuts Code POST Method:-



The most difficult part faced in Apple shortcuts was to make a proper POST request which would send the value to Adafruit IO, usually people just make use of a 3rd party service called IFTTT to get a webhook which actually behind the scenes perform a post request however the correct Api format to make a https based POST request via Apple shortcuts in unavailable on the internet which seemed to be a small but great achievement to completely get rid of any 3rd party service especially IFTTT.

9) Circuit Python Code and Adafruit IO Dashboard

The Board was programmed using Circuit python, there were a few difficulties where the code was unresponsive and was required to be nuked with the flash_nuke.uf2 file.
The editor that was widely used for this project was the Mu Editor and sometimes Visual Studio Code seemed easier for search. Below is the code used in the project:-

```
1.  # Albin's Code
2.
3.  import board
4.  import busio
5.  import digitalio
6.  import time
7.  from random import randint
8.  from adafruit_wiznet5k.adafruit_wiznet5k import *
9.  import adafruit_wiznet5k.adafruit_wiznet5k_socket as socket
10.
11. from adafruit_io.adafruit_io import IO_MQTT
12. import adafruit_minimqtt.adafruit_minimqtt as MQTT
13. from secrets import secrets
14.
15. from time import sleep
16.
17. # ////////////Display Import\\\\\\\\
18.
19. import displayio
20. import terminalio
21. import adafruit_displayio_ssd1306
22. from adafruit_display_text import label
23.
24. # ////////////////////////////////////
25.
26. displayio.release_displays()
27.
28. # ///////////////LIVE SERIAL DISPLAY///////////////
29. i2c = busio.I2C (scl=board.GP1, sda=board.GP0) # This RPi Pico way to call I2C
30. display_bus = displayio.I2CDisplay (i2c, device_address = 0x3C) # The address of my
    Board
31. display = adafruit_displayio_ssd1306.SSD1306(display_bus, width=128, height=55)
32. # ///////////////LIVE SERIAL DISPLAY///////////////
33.
34. # Set your Adafruit IO Username and Key in secrets.py
35.
36. aio_username = secrets["aio_username"]
37. aio_key = secrets["aio_key"]
38.
39. ##SPI
40. SPI0_SCK = board.GP18
41. SPI0_TX = board.GP19
42. SPI0_RX = board.GP16
```

```
SPI0_CSn = board.GP17

#Reset
W5x00_RSTn = board.GP20

print("  ""HOME AUTOMATION" "\n"    "  WIZNET CONTEST" "\n" "      " "2022")
sleep(4)
print("\n" "   Created by" "\n"  "                    " "Albin Joseph")
sleep(4)

print("Pinging Adafruit             IO""\n"" Wiznet5k (DHCP)")
# Setup your network configuration below
# random MAC, later should change this value on your vendor ID
MY_MAC = (0x00, 0x01, 0x02, 0x03, 0x04, 0x05)
IP_ADDRESS = (192, 168, 1, 100)
SUBNET_MASK = (255, 255, 255, 0)
GATEWAY_ADDRESS = (192, 168, 1, 1)
DNS_SERVER = (8, 8, 8, 8)

ethernetRst = digitalio.DigitalInOut(W5x00_RSTn)
ethernetRst.direction = digitalio.Direction.OUTPUT

# For Adafruit Ethernet FeatherWing
cs = digitalio.DigitalInOut(SPI0_CSn)
# For Particle Ethernet FeatherWing
# cs = digitalio.DigitalInOut(board.D5)

spi_bus = busio.SPI(SPI0_SCK, MOSI=SPI0_TX, MISO=SPI0_RX)

# Reset W5x00 first
ethernetRst.value = False
time.sleep(1)
ethernetRst.value = True

# Initialize ethernet interface with DHCP
eth = WIZNET5K(spi_bus, cs, is_dhcp=True, mac=MY_MAC, debug=False)

print("Chip Version:", eth.chip)
print("MAC Address:", [hex(i) for i in eth.mac_address])
print("My IP address is:", eth.pretty_ip(eth.ip_address))

### Code ###
# Define callback methods which are called when events occur
# pylint: disable=unused-argument, redefined-outer-name
def connected(clinet):
    # This function will be called when the mqtt_client is connected
    # successfully to the broker.
    print("Connected to Adafruit IO!")

    # Subscribe to Group
    io.subscribe(group_key=group_name)

def disconnected(clinet):
    # This method is called when the mqtt_client disconnects
    # from the broker.
    print("Disconnected from Adafruit IO!")

 def subscribe(client, userdata, topic, granted_qos):
     # This method is called when the client subscribes to a new feed.
     print("Subscribed to {0} with QOS level {1}".format(topic, granted_qos))

 def message(client, topic, message):
     # Method called when a client's subscribed feed has a new value.
     print(" {0}: {1}".format(topic, message))

 # Board LED CONTROL
////////////////////////////////////////////////////////////////////////////////
 BoardLed = digitalio.DigitalInOut(board.GP25)
 BoardLed.direction = digitalio.Direction.OUTPUT

```

```
112.  def on_BoardLed_msg(client, topic, message):
113.      # Method called when a client's subscribed feed has a new value.
114.      print(" {0}: {1}".format(topic, message))
115.      if message == "on":
116.          BoardLed.value = True
117.      elif message == "off":
118.          BoardLed.value = False
119.      else:
120.          print("Unexpected message on BoardLed feed")
121.
122.  # External LED CONTROL
    ////////////////////////////////////////////////////////////////////////////////
123.  ExternalLed = digitalio.DigitalInOut(board.GP2)
124.  ExternalLed.direction = digitalio.Direction.OUTPUT
125.
126.  def on_ExternalLed_msg(client, topic, message):
127.      # Method called when a client's subscribed feed has a new value.
128.      print(" {0}: {1}".format(topic, message))
129.      if message == "on":
130.          ExternalLed.value = True
131.      elif message == "off":
132.          ExternalLed.value = False
133.      else:
134.          print("Unexpected message on ExternalLed feed")
135.
136.  # KITCHEN FAN CONTROL
    ////////////////////////////////////////////////////////////////////////////////
137.  KitchenFan = digitalio.DigitalInOut(board.GP7)
138.  KitchenFan.direction = digitalio.Direction.OUTPUT
139.
140.  def on_KitchenFan_msg(client, topic, message):
141.      # Method called when a client's subscribed feed has a new value.
142.      print(" {0}: {1}".format(topic, message))
143.      if message == "on":
144.          KitchenFan.value = True
145.      elif message == "off":
146.          KitchenFan.value = False
147.      else:
148.          print("Unexpected message on KitchenFan feed")
149.
150.  # BEDROOM FAN CONTROL
    ////////////////////////////////////////////////////////////////////////////////
151.  BedroomFan = digitalio.DigitalInOut(board.GP6)
152.  BedroomFan.direction = digitalio.Direction.OUTPUT
153.
154.  def on_BedroomFan_msg(client, topic, message):
155.      # Method called when a client's subscribed feed has a new value.
156.      print(" {0}: {1}".format(topic, message))
157.      if message == "on":
158.          BedroomFan.value = True
159.      elif message == "off":
160.          BedroomFan.value = False
161.      else:
162.          print("Unexpected message on BedroomFan feed")
163.
164.  # HALL FAN CONTROL GPIO PIN CONTROL
    ////////////////////////////////////////////////////////////////////////////////
165.
166.  # HallFan = digitalio.DigitalInOut(board.GP22)
167.  # HallFan.direction = digitalio.Direction.OUTPUT
168.
169.  # def on_HallFan_msg(client, topic, message):
170.  #    Method called when a client's subscribed feed has a new value.
171.  #     print(" {0}: {1}".format(topic, message))
172.  #     if message == "on":
173.  #         HallFan.value = True
174.  #     elif message == "off":
175.  #         HallFan.value = False
176.  #     else:
177.  #         print("Unexpected message on HallFan feed")
```

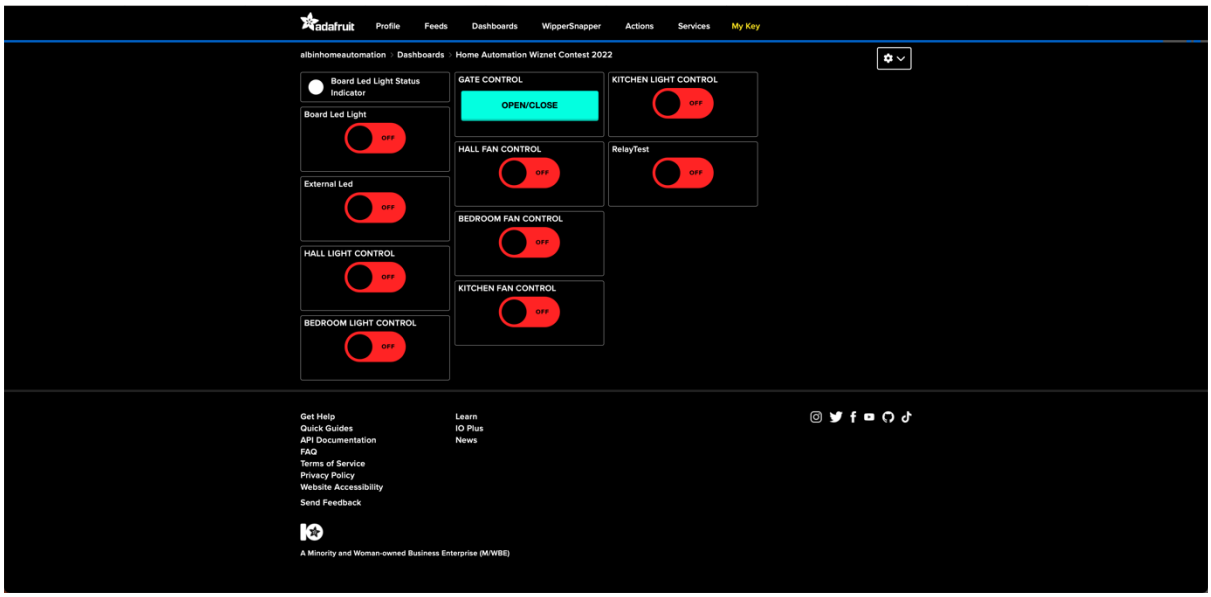
```
178.
179.  #
    ////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////
180.
181.  # Relay CONTROL System (Low/False is ON
    )//////////////////////////////////////////////////////////////////////////////////////
182.
183.  # /////// RelayTest \\\\\\\
184.
185.  RelayTest = digitalio.DigitalInOut(board.GP9)
186.  RelayTest.direction = digitalio.Direction.OUTPUT
187.  RelayTest.value = True
188.
189.  def on_RelayTest_msg(client, topic, message):
190.      # Method called when a client's subscribed feed has a new value.
191.      print(" {0}: {1}".format(topic, message))
192.      if message == "on":
193.          RelayTest.value = False
194.      elif message == "off":
195.          RelayTest.value = True
196.      else:
197.          print("Unexpected message on RelayTest feed")
198.
199.  # /////// HALL FAN CONTROL (RELAY) \\\\\\\\
200.
201.  HallFan = digitalio.DigitalInOut(board.GP15)
202.  HallFan.direction = digitalio.Direction.OUTPUT
203.  HallFan.value = True
204.
205.  def on_HallFan_msg(client, topic, message):
206.  ##  Method called when a client's subscribed feed has a new value.
207.      print(" {0}: {1}".format(topic, message))
208.      if message == "on":
209.          HallFan.value = False
210.      elif message == "off":
211.          HallFan.value = True
212.      else:
213.          print("Unexpected message on HallFan feed")
214.
215.  # HallFanRelay = digitalio.DigitalInOut(board.GP15)
216.  # HallFanRelay.direction = digitalio.Direction.OUTPUT
217.  # HallFanRelay.value = True
218.
219.  # def on_HallFanRelay_msg(client, topic, message):
220.  ##  Method called when a client's subscribed feed has a new value.
221.  #     print(" {0}: {1}".format(topic, message))
222.  #     if message == "on":
223.  #         HallFanRelay.value = False
224.  #     elif message == "off":
225.  #         HallFanRelay.value = True
226.  #     else:
227.  #         print("Unexpected message on HallFanRelay feed")
228.
229.  # /////// HALL LIGHT \\\\\\\
230.
231.  HallLight = digitalio.DigitalInOut(board.GP14)
232.  HallLight.direction = digitalio.Direction.OUTPUT
233.  HallLight.value = True
234.
235.  def on_HallLight_msg(client, topic, message):
236.      # Method called when a client's subscribed feed has a new value.
237.      print(" {0}: {1}".format(topic, message))
238.      if message == "on":
239.          HallLight.value = False
240.      elif message == "off":
241.          HallLight.value = True
242.      else:
243.          print("Unexpected message on HallLight feed")
244.
```

```
# /////// BEDROOM LIGHT \\\\\\\

BedroomLight = digitalio.DigitalInOut(board.GP13)
BedroomLight.direction = digitalio.Direction.OUTPUT
BedroomLight.value = True

def on_BedroomLight_msg(client, topic, message):
    # Method called when a client's subscribed feed has a new value.
    print(" {0}: {1}".format(topic, message))
    if message == "on":
        BedroomLight.value = False
    elif message == "off":
        BedroomLight.value = True
    else:
        print("Unexpected message on BedroomLight feed")

# /////// KITCHEN LIGHT \\\\\\\

KitchenLight = digitalio.DigitalInOut(board.GP12)
KitchenLight.direction = digitalio.Direction.OUTPUT
KitchenLight.value = True

def on_KitchenLight_msg(client, topic, message):
    # Method called when a client's subscribed feed has a new value.
    print(" {0}: {1}".format(topic, message))
    if message == "on":
        KitchenLight.value = False
    elif message == "off":
        KitchenLight.value = True
    else:
        print("Unexpected message on KitchenLight feed")

# ************GATE AUTOMATION************
# /////// GateButton \\\\\\\

GateButton = digitalio.DigitalInOut(board.GP8)
GateButton.direction = digitalio.Direction.OUTPUT
GateButton.value = True

def on_GateButton_msg(client, topic, message):
    # Method called when a client's subscribed feed has a new value.
    print(" {0}: {1}".format(topic, message))
    if message == "on":
        GateButton.value = False
        sleep(1)
        GateButton.value = True
        print("Gate triggered!!")
        message == "off"
    else:
        print("GATE Triggered" "\n" "with Adafruit IO" "\n" "    " "!!")

# ////////////////////////////////////////////////////////////////////////////////

# Initialize MQTT interface with the ethernet interface
MQTT.set_socket(socket, eth)

# Initialize a new MQTT Client object
mqtt_client = MQTT.MQTT(
    broker="io.adafruit.com",
    username=secrets["aio_username"],
    password=secrets["aio_key"],
    is_ssl=False,
)

# Initialize an Adafruit IO MQTT Client
io = IO_MQTT(mqtt_client)

# Setup the callback methods above
io.on_connect = connected
io.on_disconnect = disconnected
```

```
315.  io.on_message = message
316.  io.on_subscribe = subscribe
317.
318.  # Set up a callback for the BoardLed feed ////////////////////////////////////////
319.  io.add_feed_callback("BoardLed", on_BoardLed_msg)
320.
321.  # Set up a callback for the ExternalLed feed
    ////////////////////////////////////////
322.  io.add_feed_callback("ExternalLed", on_ExternalLed_msg)
323.
324.  # Set up a callback for the KitchenFan feed ////////////////////////////////////////
325.  io.add_feed_callback("KitchenFan", on_KitchenFan_msg)
326.
327.  # Set up a callback for the BedroomFan feed ////////////////////////////////////////
328.  io.add_feed_callback("BedroomFan", on_BedroomFan_msg)
329.
330.  # Set up a callback for the HallFan feed ////////////////////////////////////////
331.  io.add_feed_callback("HallFan", on_HallFan_msg)
332.
333.  # ////////////////////////////////////////////////////////////////////////////////
334.
335.  # Set up a callback for the RelayTest feed ////////////////////////////////////////
336.  io.add_feed_callback("RelayTest", on_RelayTest_msg)
337.
338.  ## Set up a callback for the HallFanRelay feed
    ////////////////////////////////////////
339.  # io.add_feed_callback("HallFanRelay", on_HallFanRelay_msg)
340.
341.  # Set up a callback for the HallLight feed ////////////////////////////////////////
342.  io.add_feed_callback("HallLight", on_HallLight_msg)
343.
344.  # Set up a callback for the BedroomLight feed
    ////////////////////////////////////////
345.  io.add_feed_callback("BedroomLight", on_BedroomLight_msg)
346.
347.  # Set up a callback for the KitchenLight feed
    ////////////////////////////////////////
348.  io.add_feed_callback("KitchenLight", on_KitchenLight_msg)
349.
350.  # Set up a callback for the GateButton feed ////////////////////////////////////////
351.  io.add_feed_callback("GateButton", on_GateButton_msg)
352.
353.  ## Group name
354.  group_name = "weatherstation" # Comenting Throws error need to check
355.
356.  ## Feeds within the group
357.  temp_feed = "weatherstation.temperature" # omenting Throws error need to check
358.  humid_feed = "weatherstation.humidity"  # Comenting Throws error need to check
359.
360.  # Connect to Adafruit IO
361.  print("Connecting to Adafruit IO...")
362.  io.connect()
363.
364.  # ////////////////////////////ADD ON
    SUBSCRIPTION////////////////////////////////////////////////////////
365.
366.  # # Subscribe to all messages on the BoardLed feed
367.  io.subscribe("BoardLed")
368.
369.  # # Subscribe to all messages on the ExternalLed feed
370.  io.subscribe("ExternalLed")
371.
372.  # Subscribe to all messages on the KitchenFan feed
373.  io.subscribe("KitchenFan")
374.
375.  # Subscribe to all messages on the BedroomFan feed
376.  io.subscribe("BedroomFan")
377.
378.  # # Subscribe to all messages on the HallFan feed
379.  io.subscribe("HallFan")
```

```
380.
381.  # ////////////////////////////RELAY
    SUBSCRIPTION//////////////////////////////////////////////////////////
382.
383.  # # Subscribe to all messages on the RelayTest feed
384.  io.subscribe("RelayTest")
385.
386.  # Subscribe to all messages on the HallFanRelay feed
387.  # io.subscribe("HallFanRelay")
388.
389.  # # Subscribe to all messages on the HallLight feed
390.  io.subscribe("HallLight")
391.
392.  # # Subscribe to all messages on the BedroomLight feed
393.  io.subscribe("BedroomLight")
394.
395.  # # Subscribe to all messages on the KitchenLight feed
396.  io.subscribe("KitchenLight")
397.
398.  # # Subscribe to all messages on the GateButton feed
399.  io.subscribe("GateButton")
400.
401.  # ////////////////////////////////////////////////////////////////////////////////////////
402.
403.  print("Connected to Adafruit IO")
404.  print("    HandShake         SUCCESS!!")
405.  print("WAITING FOR INPUT")
406.
407.  while True:
408.      io.loop()
409.
```

Adafruit IO Dashboard:-



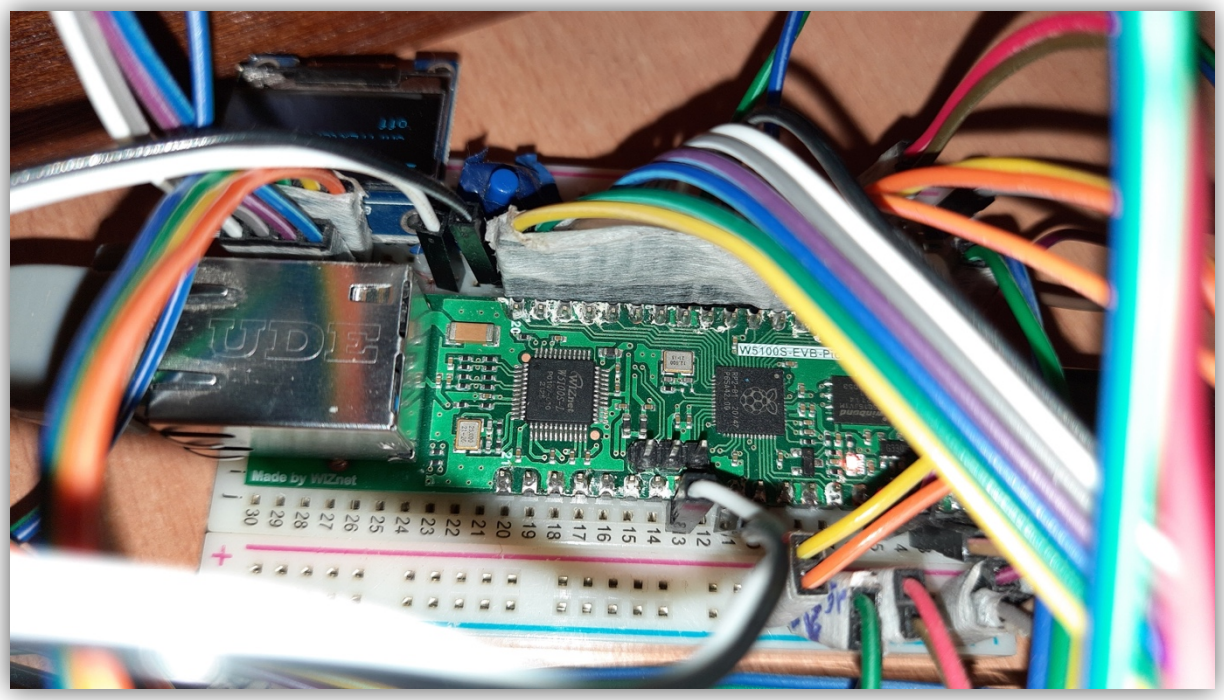
10) Results and Conclusion

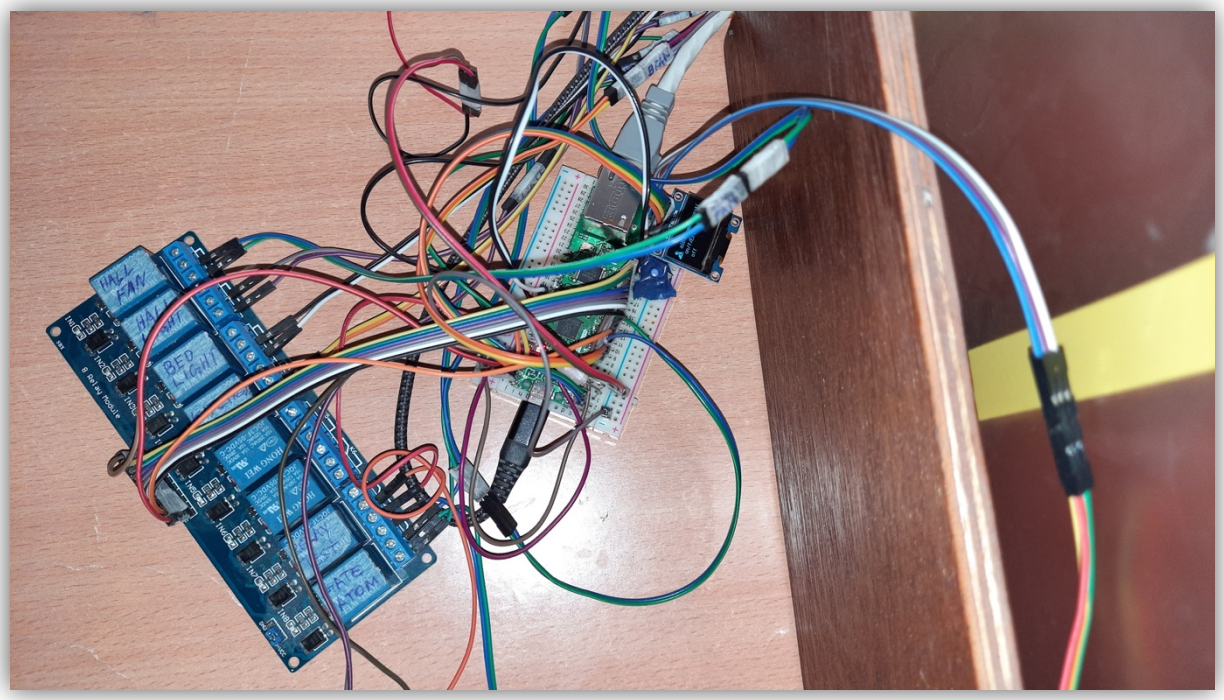
Well the results were great! Automated gates lights and fans all powered by the lone board W5100S-EVB-Pico programmed with circuit python, connected to Adafruit IO which was a possibility only because of Wiznet's brilliant idea of adding an
Ethernet (RJ45) port powered by the great W5100S hardwired TCP/IP chip and equally important wiznet 5k ethernet library and adafruit library which inspired and helped me complete this amazing project successfully.

I was also able to learn and gain a lot of experience from this project and I'm really grateful for this grand opportunity wiznet provided to us.

<u>Please Note:-</u>
A video on this project including a working demo is being prepared which will be shared soon.

The below images in the following pages shows some images that I have captured :-

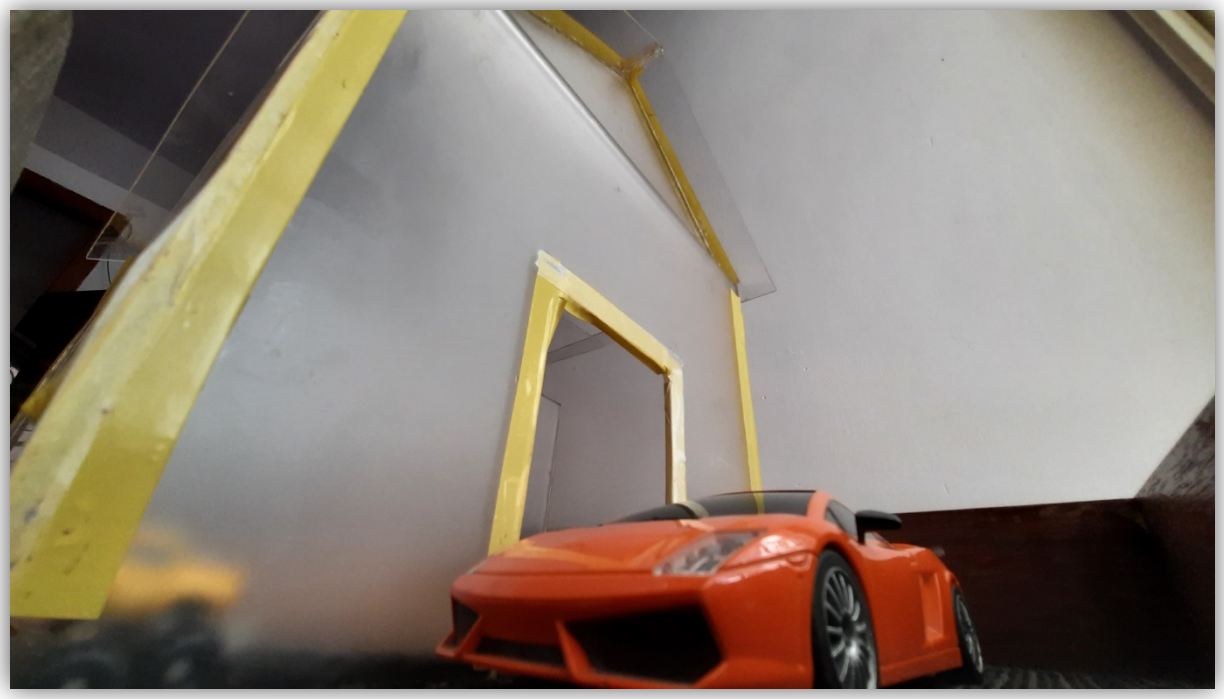

Made by WIZnet
W5100S-EVB-Pi


HALL FAN
BED LIGHT

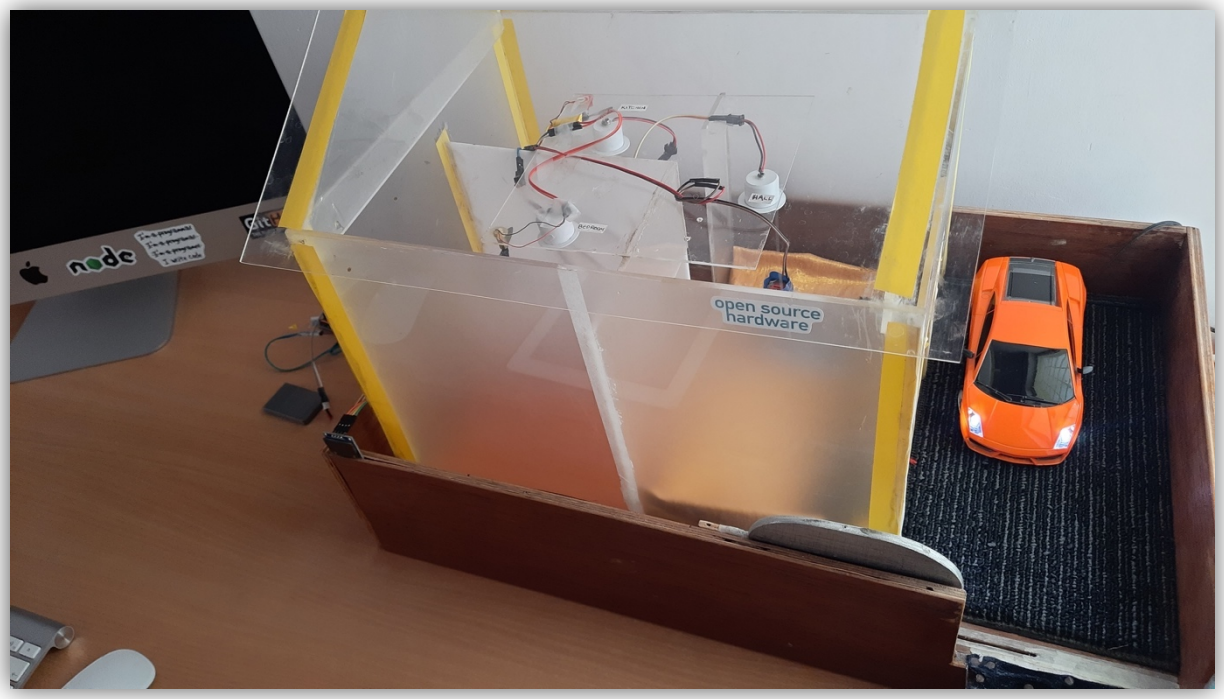

open source
hardware


open source
hardware

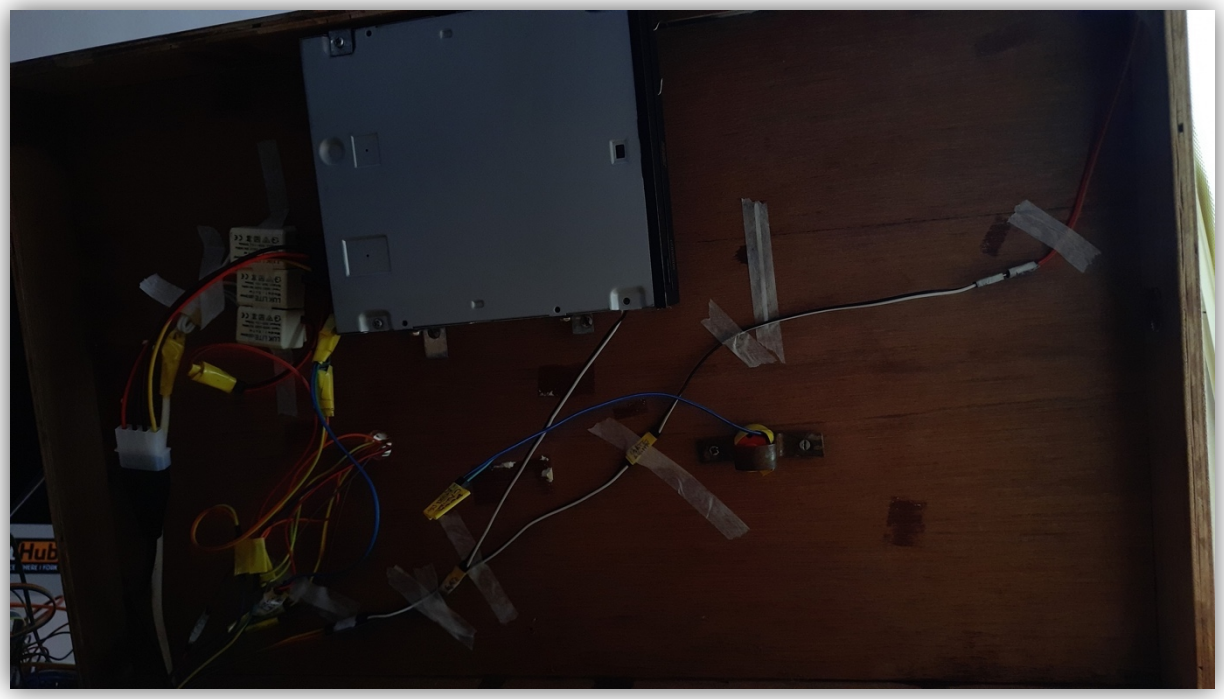

GND VCC SCL SDA
Created by
Albin Joseph

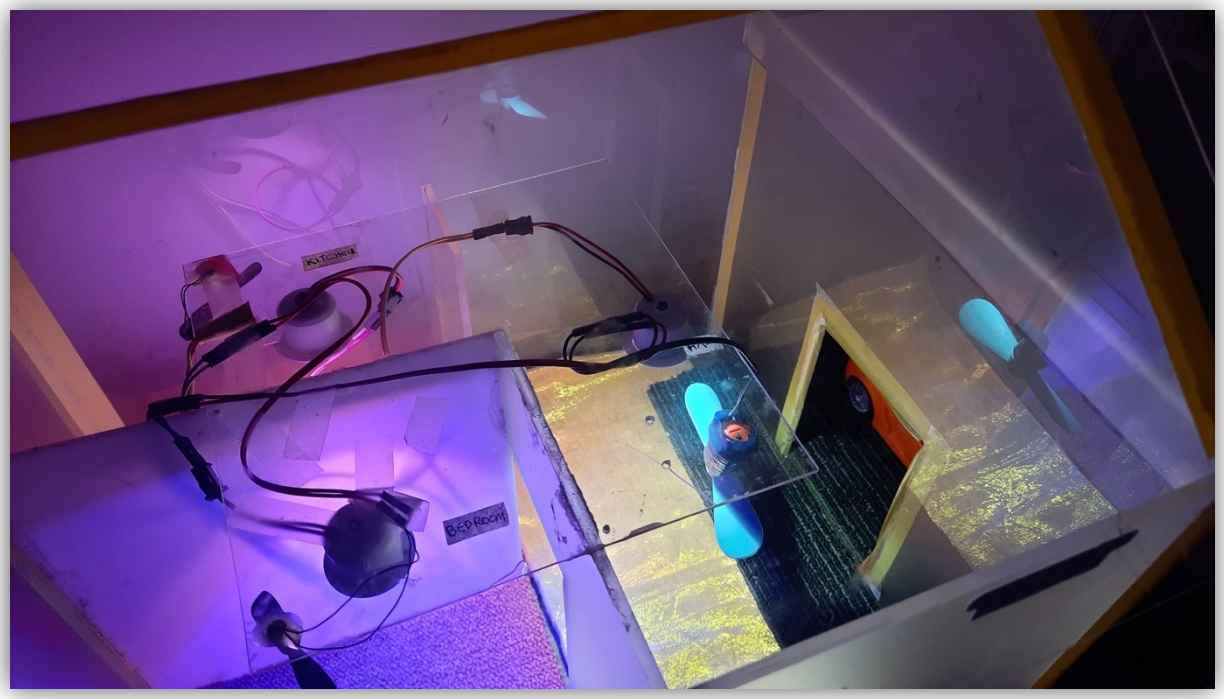


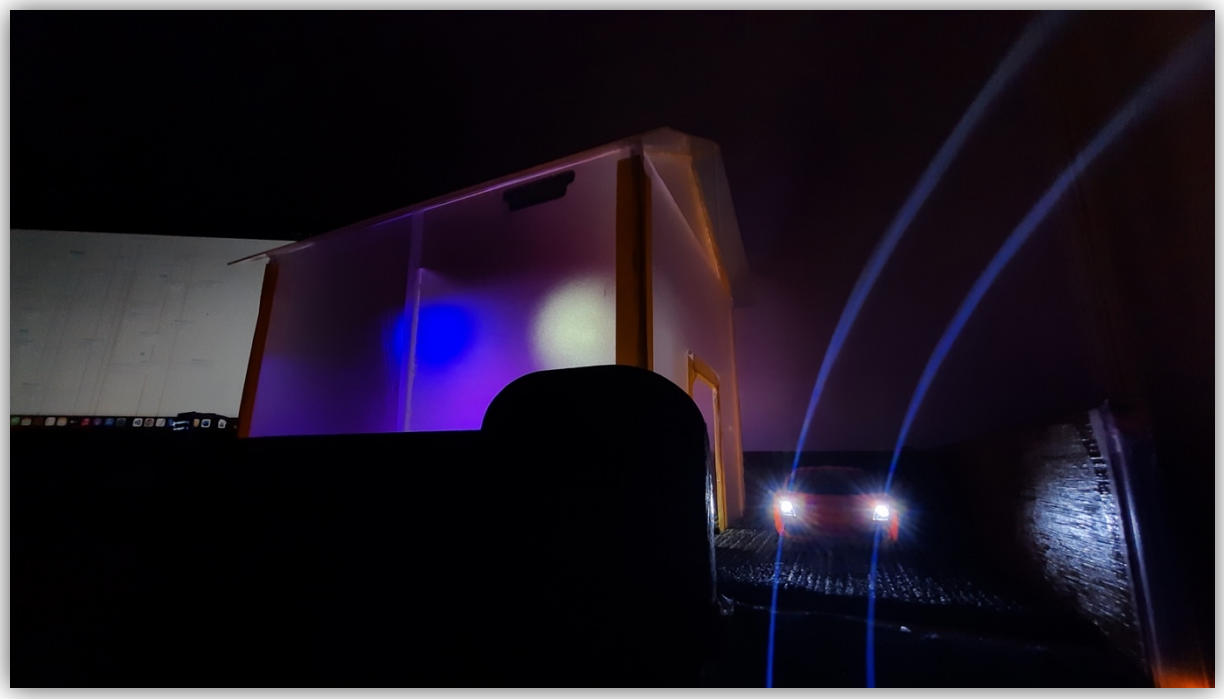

KITCHEN
HALL
BEDROOM

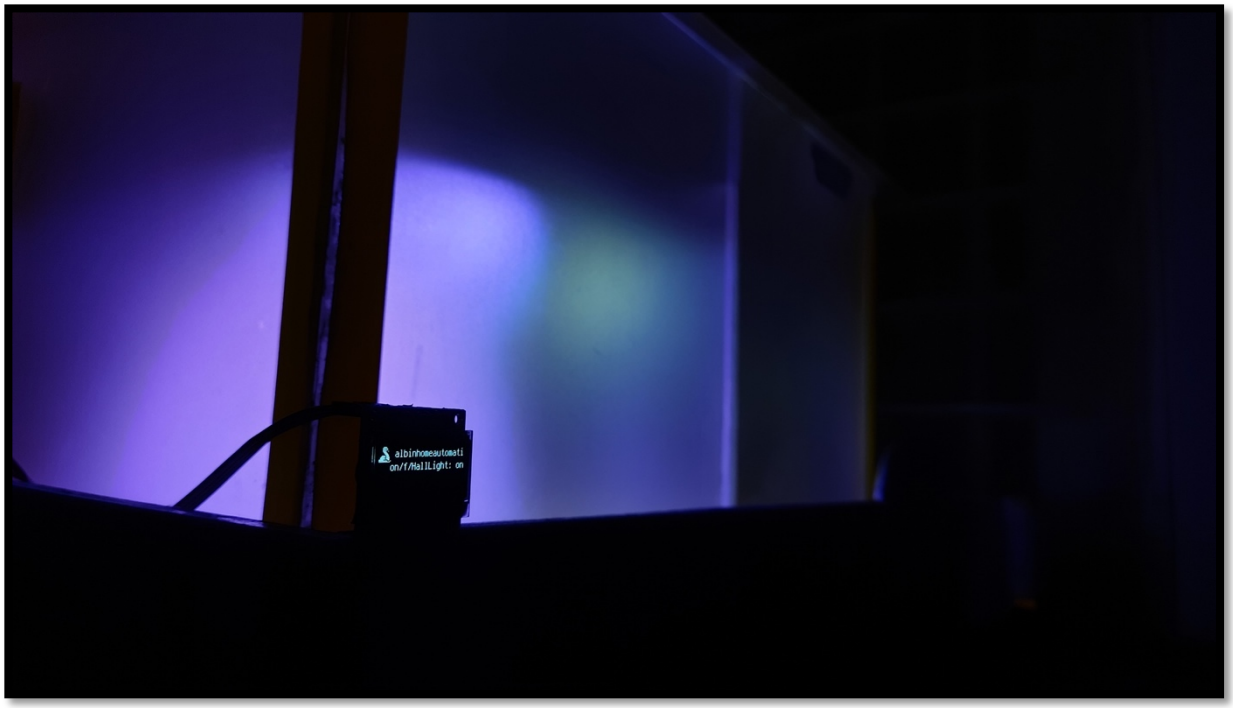

node
open source
hardware


HALL




KITCHEN
BEDROOM




albinhomeautomati
on/f/HallLight: on

THANK YOU

END OF DOCUMENT