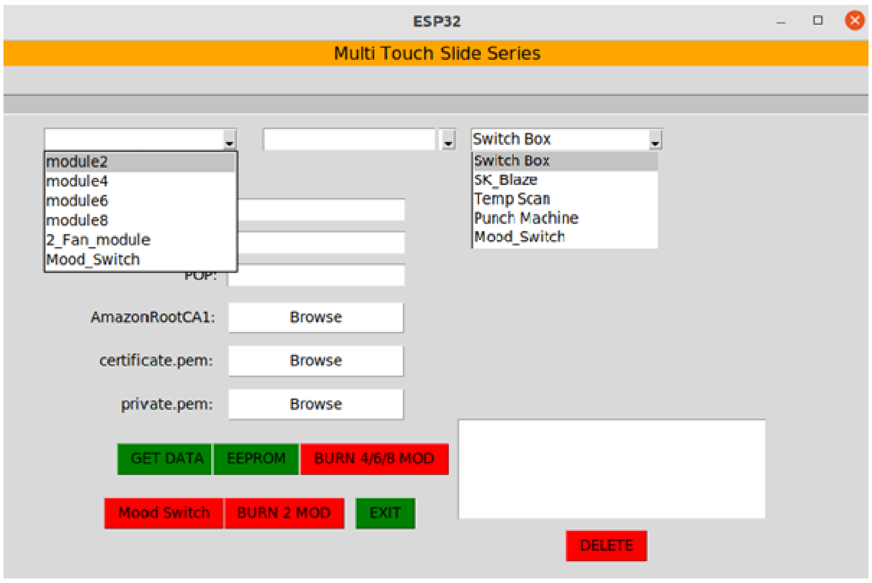
# ESP32 Production Code and GUI Documentation



- **ESP32 Production GUI** is used for the programming the Skroman switch devices in a specific way particular unique id for each and individual devices.
- The above screen shows the dropdown options of different devices, ESP entry box, Unique ID entry box, POP, certificates like **AmazonRootCA1**, **certificate.pem** and **private.pem** which needs to be browsed from a stored specific folder.
- These certificates are generated new and unique to each client or each device to be programmed based on the requirement and maintained in a separate folder according to the name of the client.
- To program a code, first select a value from first left dropdown of which device you are going to program, i.e.,

1. **Module 2**
   - 20000
   - 23000
2. **Module 4**
   - 44010
   - 45000
   - 46000
3. **Module 6**

- 65010
- 67000

4. **Module 8**

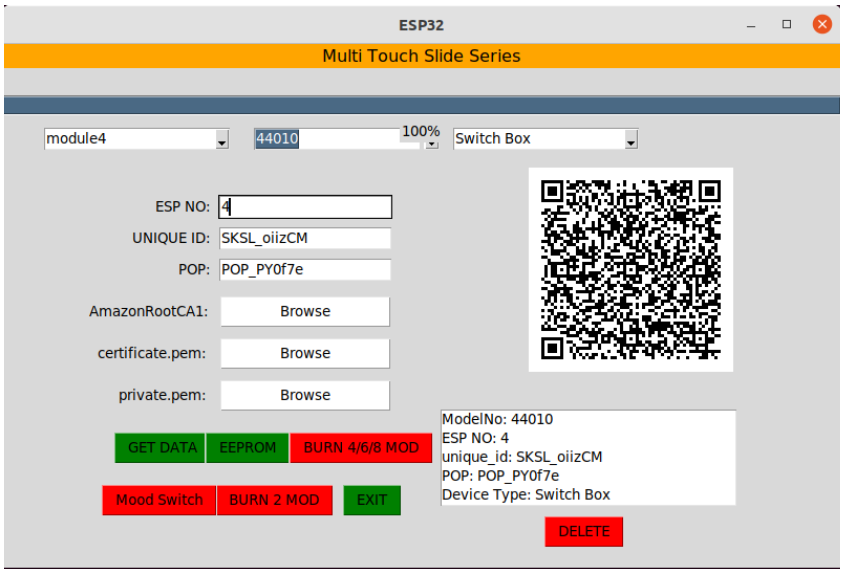
- 87010
- 89000

5. **2 Fan Module**

- 88020

6. **Mood Switch**

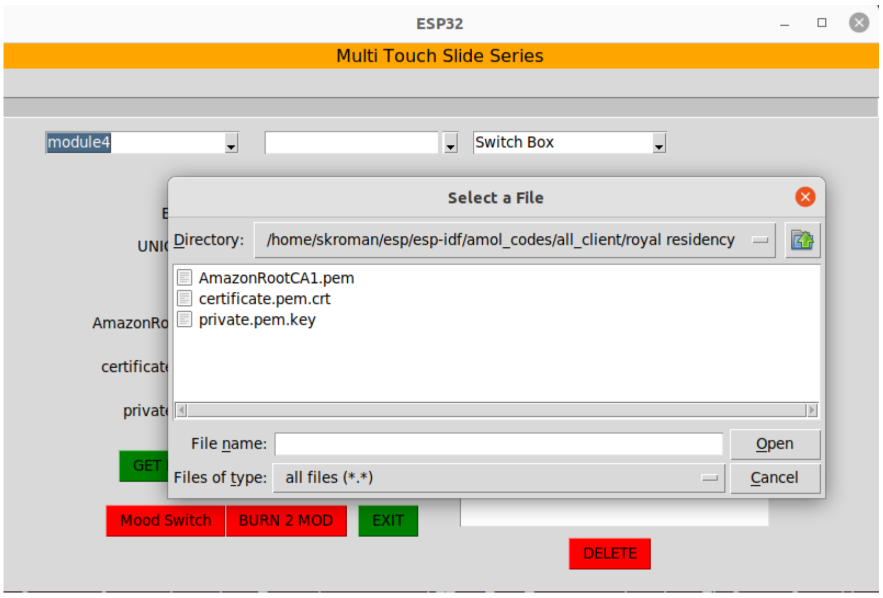
- Mood Switch



ModelNo: 44010
ESP NO: 4
unique_id: SKSL_oiizCM
POP: POP_PY0f7e
Device Type: Switch Box

- After selecting the values from both dropdowns, we need to select the mod of the device from the extreme right dropdown, i.e., **Switch box, Mood Switch, Punch Machine, Sk Blaze, or Temp scan**. For default it has been set as **Switch box** for the values of Modules 2, 4, 6, 8.
- In the beginning, we need the select the module values and the module type from dropdown as per the module specification to program the device.
- After selecting the module values, enter the track recorded ESP NO and click the **GET DATA**, by this, we`ll receive the values of UNIQUE ID and POP from the server which is on the **AWS** instance.
- After the values are entered, we need to browse and attach three certificates; those are **AmazonRootCA1**, **certificate.pem** and **private.pem.**

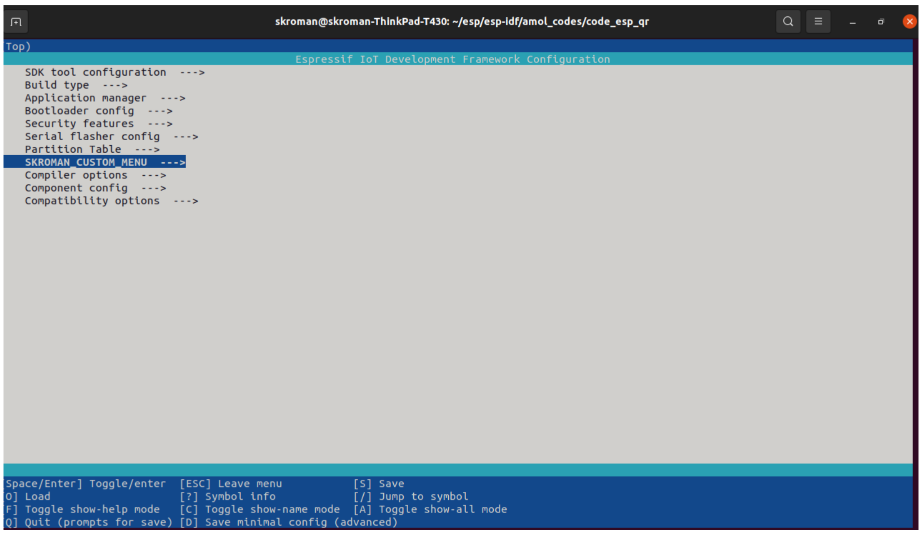
- These certificates are unique for each individual client, so hence we need to select their respective certificates for their devices to program.



Directory: /home/skroman/esp/esp-idf/amol_codes/all_client/royal residency

```
AmazonRootCA1.pem
certificate.pem.crt
private.pem.key
```

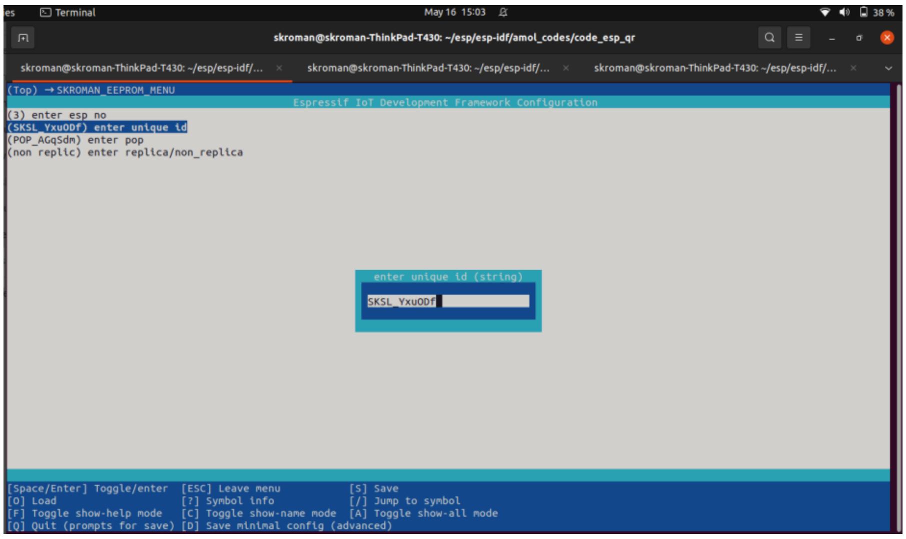
- Later then after selecting the all three certificates from a specified client`s folder, we need to EEPROM the code by clicking on the **EEPROM** button. EEPROM code is used to full clean and rebuild the code in the device for new fresh entries of the code.
- As the EEPROM button is clicked, come on the terminal page, where you can see the programming process happening. Immediately a window will arise which is an **ESP-IDF** window, where we need to select the option **SKROMAN_CUSTOM_MENU.**



```
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/amol_codes/code_esp_qr
Top)
                    Espressif IoT Development Framework Configuration
   SDK tool configuration  --->
   Build type  --->
   Application manager  --->
   Bootloader config  --->
   Security features  --->
   Serial flasher config  --->
   Partition Table  --->
   SKROMAN_CUSTOM_MENU  --->
   Compiler options  --->
   Component config  --->
   Compatibility options  --->

Space/Enter] Toggle/enter  [ESC] Leave menu          [S] Save
O] Load                    [?] Symbol info           [/] Jump to symbol
F] Toggle show-help mode   [C] Toggle show-name mode [A] Toggle show-all mode
Q] Quit (prompts for save) [D] Save minimal config (advanced)
```

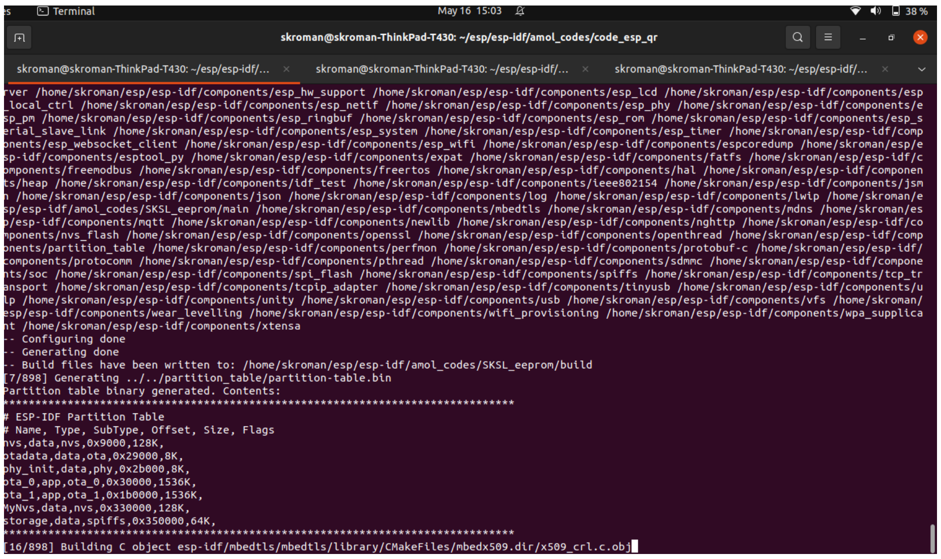
- After entering the option, give the values of
  - ESP

    - UNIQUE ID
    - POP
    - REPLICA/NON-REPLICA
- Enter each values and then press ‘s’ button to save, after saving escape back to the terminal page, press escape button till it comes to the terminal page.


Terminal
May 16 15:03
38 %
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/amol_codes/code_esp_qr
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/...
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/...
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/...

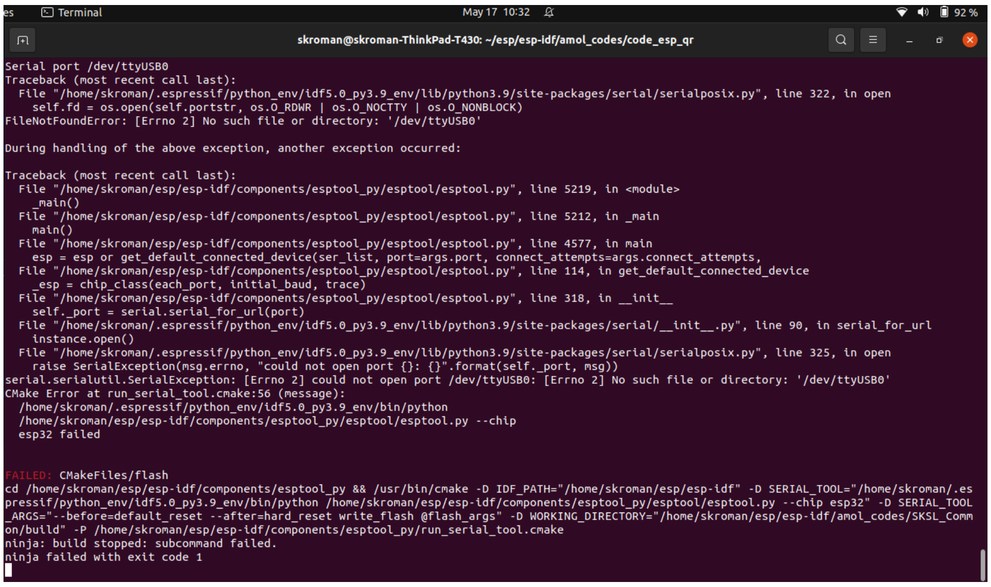
```
(Top) → SKROMAN_EEPROM_MENU
                Espressif IoT Development Framework Configuration
(3) enter esp no
(SKSL_YxuODf) enter unique id
(POP_AGqSdm) enter pop
(non replic) enter replica/non_replica

                      enter unique id (string)
                      SKSL_YxuODf

[Space/Enter] Toggle/enter   [ESC] Leave menu           [S] Save
[O] Load                     [?] Symbol info            [/] Jump to symbol
[F] Toggle show-help mode    [C] Toggle show-name mode  [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)
```

- After coming back to the terminal you can see that the programming process in a seek (0/898…).


Terminal
May 16 15:03
38 %
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/amol_codes/code_esp_qr
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/...
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/...
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/...

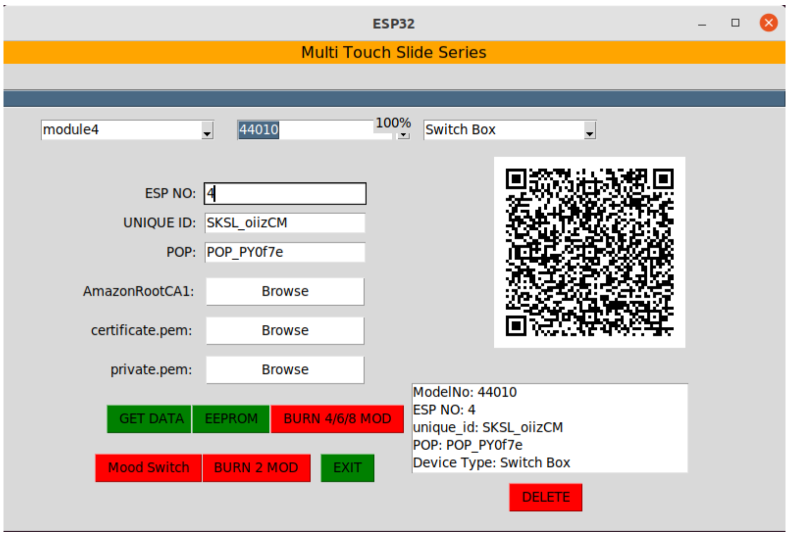
```
rver /home/skroman/esp/esp-idf/components/esp_hw_support /home/skroman/esp/esp-idf/components/esp_lcd /home/skroman/esp/esp-idf/components/esp
_local_ctrl /home/skroman/esp/esp-idf/components/esp_netif /home/skroman/esp/esp-idf/components/esp_phy /home/skroman/esp/esp-idf/components/e
sp_pm /home/skroman/esp/esp-idf/components/esp_ringbuf /home/skroman/esp/esp-idf/components/esp_rom /home/skroman/esp/esp-idf/components/esp_s
erial_slave_link /home/skroman/esp/esp-idf/components/esp_system /home/skroman/esp/esp-idf/components/esp_timer /home/skroman/esp/esp-idf/comp
onents/esp_websocket_client /home/skroman/esp/esp-idf/components/esp_wifi /home/skroman/esp/esp-idf/components/espcoredump /home/skroman/esp/e
sp-idf/components/esptool_py /home/skroman/esp/esp-idf/components/expat /home/skroman/esp/esp-idf/components/fatfs /home/skroman/esp/esp-idf/c
omponents/freemodbus /home/skroman/esp/esp-idf/components/freertos /home/skroman/esp/esp-idf/components/hal /home/skroman/esp/esp-idf/componen
ts/heap /home/skroman/esp/esp-idf/components/idf_test /home/skroman/esp/esp-idf/components/ieee802154 /home/skroman/esp/esp-idf/components/jsm
n /home/skroman/esp/esp-idf/components/json /home/skroman/esp/esp-idf/components/log /home/skroman/esp/esp-idf/components/lwip /home/skroman/e
sp/esp-idf/amol_codes/SKSL_eeprom/main /home/skroman/esp/esp-idf/components/mbedtls /home/skroman/esp/esp-idf/components/mdns /home/skroman/es
p/esp-idf/components/mqtt /home/skroman/esp/esp-idf/components/newlib /home/skroman/esp/esp-idf/components/nghttp /home/skroman/esp/esp-idf/co
mponents/nvs_flash /home/skroman/esp/esp-idf/components/openssl /home/skroman/esp/esp-idf/components/openthread /home/skroman/esp/esp-idf/comp
onents/partition_table /home/skroman/esp/esp-idf/components/perfmon /home/skroman/esp/esp-idf/components/protobuf-c /home/skroman/esp/esp-idf/
components/protocomm /home/skroman/esp/esp-idf/components/pthread /home/skroman/esp/esp-idf/components/sdmmc /home/skroman/esp/esp-idf/compone
nts/soc /home/skroman/esp/esp-idf/components/spi_flash /home/skroman/esp/esp-idf/components/spiffs /home/skroman/esp/esp-idf/components/tcp_tr
ansport /home/skroman/esp/esp-idf/components/tcpip_adapter /home/skroman/esp/esp-idf/components/tinyusb /home/skroman/esp/esp-idf/components/u
lp /home/skroman/esp/esp-idf/components/unity /home/skroman/esp/esp-idf/components/usb /home/skroman/esp/esp-idf/components/vfs /home/skroman/
esp/esp-idf/components/wear_levelling /home/skroman/esp/esp-idf/components/wifi_provisioning /home/skroman/esp/esp-idf/components/wpa_supplica
nt /home/skroman/esp/esp-idf/components/xtensa
-- Configuring done
-- Generating done
-- Build files have been written to: /home/skroman/esp/esp-idf/amol_codes/SKSL_eeprom/build
[7/898] Generating ../../partition_table/partition-table.bin
Partition table binary generated. Contents:
*******************************************************************************
# ESP-IDF Partition Table
# Name, Type, SubType, Offset, Size, Flags
nvs,data,nvs,0x9000,128K,
otadata,data,ota,0x29000,8K,
phy_init,data,phy,0x2b000,8K,
ota_0,app,ota_0,0x30000,1536K,
ota_1,app,ota_1,0x1b0000,1536K,
MyNvs,data,nvs,0x330000,128K,
storage,data,spiffs,0x350000,64K,
*******************************************************************************
[16/898] Building C object esp-idf/mbedtls/mbedtls/library/CMakeFiles/mbedx509.dir/x509_crl.c.obj
```

- As the programming process comes to end, it will show a connecting……… status, at this time we need to immediately **boot** and **enable** the device using a programmer model, if failed then we need to program once again from the beginning.


Terminal
May 17 10:32
92 %
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/amol_codes/code_esp_qr

```
Serial port /dev/ttyUSB0
Traceback (most recent call last):
  File "/home/skroman/.espressif/python_env/idf5.0_py3.9_env/lib/python3.9/site-packages/serial/serialposix.py", line 322, in open
    self.fd = os.open(self.portstr, os.O_RDWR | os.O_NOCTTY | os.O_NONBLOCK)
FileNotFoundError: [Errno 2] No such file or directory: '/dev/ttyUSB0'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/skroman/esp/esp-idf/components/esptool_py/esptool/esptool.py", line 5219, in <module>
    _main()
  File "/home/skroman/esp/esp-idf/components/esptool_py/esptool/esptool.py", line 5212, in _main
    main()
  File "/home/skroman/esp/esp-idf/components/esptool_py/esptool/esptool.py", line 4577, in main
    esp = esp or get_default_connected_device(ser_list, port=args.port, connect_attempts=args.connect_attempts,
  File "/home/skroman/esp/esp-idf/components/esptool_py/esptool/esptool.py", line 114, in get_default_connected_device
    _esp = chip_class(each_port, initial_baud, trace)
  File "/home/skroman/esp/esp-idf/components/esptool_py/esptool/esptool.py", line 318, in __init__
    self._port = serial.serial_for_url(port)
  File "/home/skroman/.espressif/python_env/idf5.0_py3.9_env/lib/python3.9/site-packages/serial/__init__.py", line 90, in serial_for_url
    instance.open()
  File "/home/skroman/.espressif/python_env/idf5.0_py3.9_env/lib/python3.9/site-packages/serial/serialposix.py", line 325, in open
    raise SerialException(msg.errno, "could not open port {}: {}".format(self._port, msg))
serial.serialutil.SerialException: [Errno 2] could not open port /dev/ttyUSB0: [Errno 2] No such file or directory: '/dev/ttyUSB0'
CMake Error at run_serial_tool.cmake:56 (message):
  /home/skroman/.espressif/python_env/idf5.0_py3.9_env/bin/python
  /home/skroman/esp/esp-idf/components/esptool_py/esptool/esptool.py --chip
  esp32 failed


FAILED: CMakeFiles/flash
cd /home/skroman/esp/esp-idf/components/esptool_py && /usr/bin/cmake -D IDF_PATH="/home/skroman/esp/esp-idf" -D SERIAL_TOOL="/home/skroman/.es
pressif/python_env/idf5.0_py3.9_env/bin/python /home/skroman/esp/esp-idf/components/esptool_py/esptool/esptool.py --chip esp32" -D SERIAL_TOOL
_ARGS="--before=default_reset --after=hard_reset write_flash @flash_args" -D WORKING_DIRECTORY="/home/skroman/esp/esp-idf/amol_codes/SKSL_Comm
on/build" -P /home/skroman/esp/esp-idf/components/esptool_py/run_serial_tool.cmake
ninja: build stopped: subcommand failed.
ninja failed with exit code 1
```

- If the EEPROM process goes well and completed, then next we need to continue with the **Burn operation.** For Burn operation press the **4/6/8 Burn** if the device model is either from 4/6/8, if the model is **23000**, then click **Burn 23000,** and if **Mood Switch**, then press **Mood Switch.**
- After clicking the burn operation button, multiple functionalities take place, like the entered values of unique-id, POP, ESP, and device module no form a **QR code**, this QR code contains all the device details entered. This QR code help in easily identifying the device details of clients.
- In the internal part of the code, the burn button is connected with the function in a command named as **generate**, this function consist of a shell script file named **burn.sh**, this file consist of the file path of SKSL_Common and different functionalities like menuconfig, full clean, build and flash monitor operations. Just by clicking the burn button this all operations are performed as they are present in burn.sh shell script file.
- Similarly the operations are performed in case of **Mod_23000** and **Mood_Switch**, only the thing is Mod_23000 button is linked with the shell script file named **burn_23000.sh** and Mood_Switch is linked with the shell script named **SKSL_mood_switch.sh** respectively.

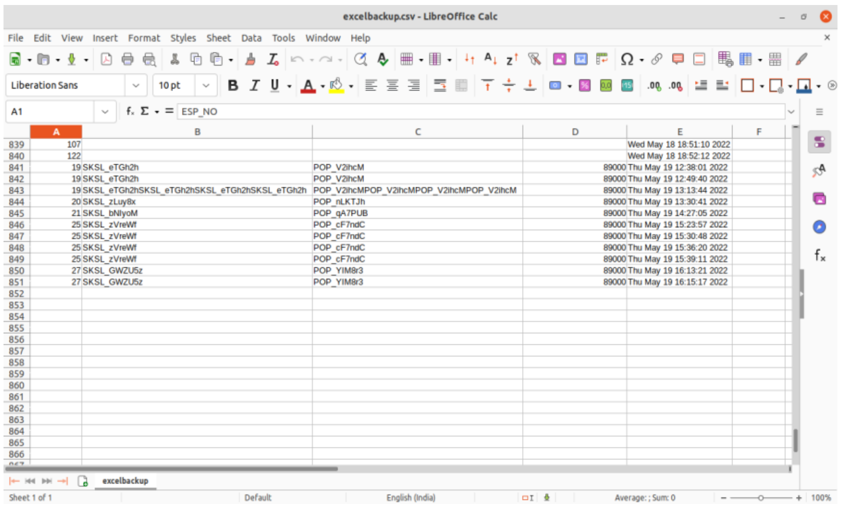


module4 | 44010 | Switch Box

ESP NO: 4
UNIQUE ID: SKSL_oiizCM
POP: POP_PY0f7e
AmazonRootCA1:
certificate.pem:
private.pem:

```
ModelNo: 44010
ESP NO: 4
unique_id: SKSL_oiizCM
POP: POP_PY0f7e
Device Type: Switch Box
```

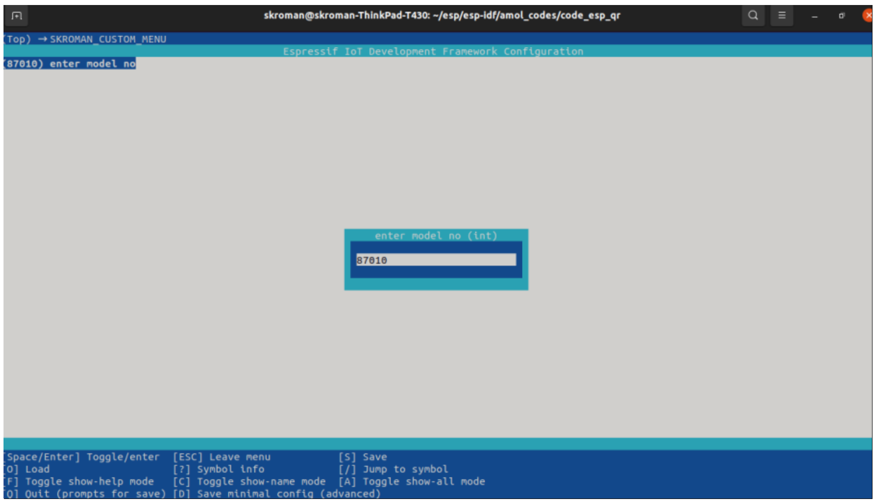
- These entered values are even stored as a backup in a **CSV file** format, which works as backup on a local system in desktop.


excelbackup.csv - LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sans
10 pt
A1
excelbackup
Sheet 1 of 1
Default
English (India)
Average: ; Sum: 0
100%

ESP_NO

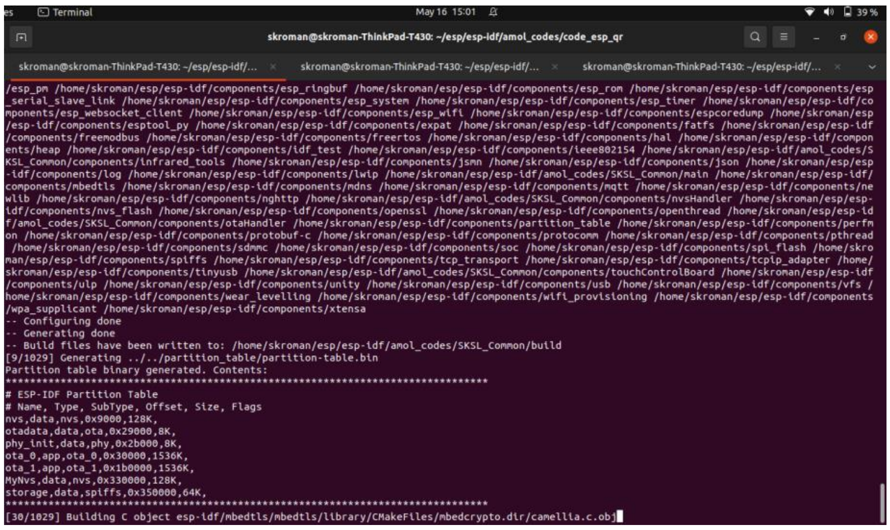
| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 839 | 107 | | | | Wed May 18 18:51:10 2022 | |
| 840 | 122 | | | | Wed May 18 18:52:12 2022 | |
| 841 | 19 | SKSL_eTGh2h | POP_V2ihcM | 89000 | Thu May 19 12:38:01 2022 | |
| 842 | 19 | SKSL_eTGh2h | POP_V2ihcM | 89000 | Thu May 19 12:49:40 2022 | |
| 843 | 19 | SKSL_eTGh2hSKSL_eTGh2hSKSL_eTGh2hSKSL_eTGh2h | POP_V2ihcMPOP_V2ihcMPOP_V2ihcMPOP_V2ihcM | 89000 | Thu May 19 13:13:44 2022 | |
| 844 | 20 | SKSL_zLuy8x | POP_nLKTJh | 89000 | Thu May 19 13:30:41 2022 | |
| 845 | 21 | SKSL_bNIyoM | POP_qA7PUB | 89000 | Thu May 19 14:27:05 2022 | |
| 846 | 25 | SKSL_zVreWf | POP_cF7ndC | 89000 | Thu May 19 15:23:57 2022 | |
| 847 | 25 | SKSL_zVreWf | POP_cF7ndC | 89000 | Thu May 19 15:30:48 2022 | |
| 848 | 25 | SKSL_zVreWf | POP_cF7ndC | 89000 | Thu May 19 15:36:20 2022 | |
| 849 | 25 | SKSL_zVreWf | POP_cF7ndC | 89000 | Thu May 19 15:39:11 2022 | |
| 850 | 27 | SKSL_GWZU5z | POP_YIM8r3 | 89000 | Thu May 19 16:13:21 2022 | |
| 851 | 27 | SKSL_GWZU5z | POP_YIM8r3 | 89000 | Thu May 19 16:15:17 2022 | |
| 852 | | | | | | |
| 853 | | | | | | |
| 854 | | | | | | |
| 855 | | | | | | |
| 856 | | | | | | |
| 857 | | | | | | |
| 858 | | | | | | |
| 859 | | | | | | |
| 860 | | | | | | |
| 861 | | | | | | |
| 862 | | | | | | |
| 863 | | | | | | |
| 864 | | | | | | |
| 865 | | | | | | |
| 866 | | | | | | |

- In burn operation after QR code and all is generated, back in the terminal, it shows all the code being programmed, after a while a windows pops-up of **ESP-IDF**, where we can set the value of model (ex- 87010) by entering into the option **SKROMAN_CUSTOM_MENU**, this is the same functional step as was in the EEPROM operation.



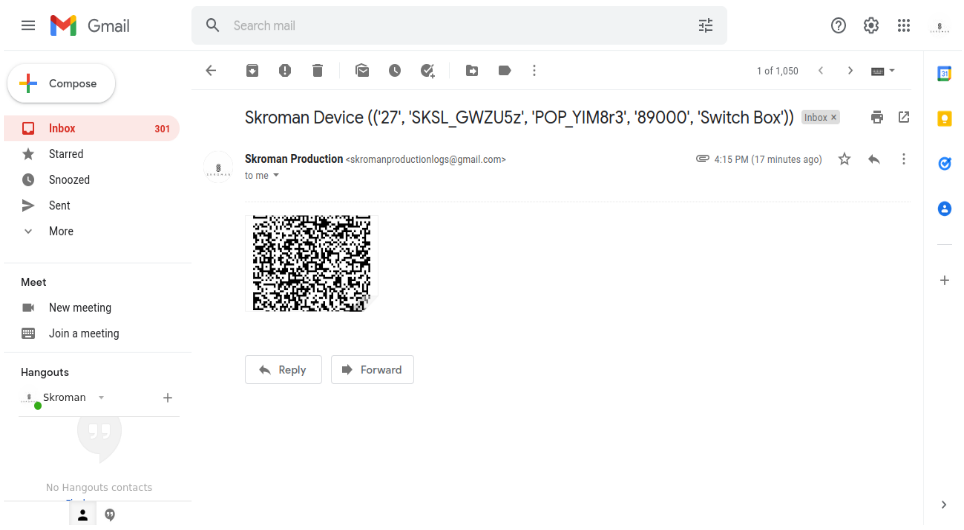
```
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/amol_codes/code_esp_qr
(Top) → SKROMAN_CUSTOM_MENU
                    Espressif IoT Development Framework Configuration
(87010) enter model no

                              enter model no (int)
                              87010

[Space/Enter] Toggle/enter   [ESC] Leave menu           [S] Save
[O] Load                     [?] Symbol info            [/] Jump to symbol
[F] Toggle show-help mode    [C] Toggle show-name mode  [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)
```

- After entering the model value, it will start processing the code programming in device which ranges as per the code (ex – 0/1039). After the code status reaches to the connecting…. Status, immediately we need to **boot** and **enable** in the programmer in order to dump the code in the device. After the code is processed and entered, we need to press the **enable** button twice again in order to get the topic values and confirmation that, the device is been programmed and burned.



```
es   Terminal                      May 16 15:01                    39 %
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/amol_codes/code_esp_qr
skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/...  skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/...  skroman@skroman-ThinkPad-T430: ~/esp/esp-idf/...
/esp_pm /home/skroman/esp/esp-idf/components/esp_ringbuf /home/skroman/esp/esp-idf/components/esp_rom /home/skroman/esp/esp-idf/components/esp
_serial_slave_link /home/skroman/esp/esp-idf/components/esp_system /home/skroman/esp/esp-idf/components/esp_timer /home/skroman/esp/esp-idf/co
mponents/esp_websocket_client /home/skroman/esp/esp-idf/components/esp_wifi /home/skroman/esp/esp-idf/components/espcoredump /home/skroman/esp
/esp-idf/components/esptool_py /home/skroman/esp/esp-idf/components/expat /home/skroman/esp/esp-idf/components/fatfs /home/skroman/esp/esp-idf
/components/freemodbus /home/skroman/esp/esp-idf/components/freertos /home/skroman/esp/esp-idf/components/hal /home/skroman/esp/esp-idf/compon
ents/heap /home/skroman/esp/esp-idf/components/idf_test /home/skroman/esp/esp-idf/components/ieee802154 /home/skroman/esp/esp-idf/amol_codes/S
KSL_Common/components/infrared_tools /home/skroman/esp/esp-idf/components/jsmn /home/skroman/esp/esp-idf/components/json /home/skroman/esp/esp
-idf/components/log /home/skroman/esp/esp-idf/components/lwip /home/skroman/esp/esp-idf/amol_codes/SKSL_Common/main /home/skroman/esp/esp-idf/
components/mbedtls /home/skroman/esp/esp-idf/components/mdns /home/skroman/esp/esp-idf/components/mqtt /home/skroman/esp/esp-idf/components/ne
wlib /home/skroman/esp/esp-idf/components/nghttp /home/skroman/esp/esp-idf/amol_codes/SKSL_Common/components/nvsHandler /home/skroman/esp/esp-
idf/components/nvs_flash /home/skroman/esp/esp-idf/components/openssl /home/skroman/esp/esp-idf/components/openthread /home/skroman/esp/esp-id
f/amol_codes/SKSL_Common/components/otaHandler /home/skroman/esp/esp-idf/components/partition_table /home/skroman/esp/esp-idf/components/perfm
on /home/skroman/esp/esp-idf/components/protobuf-c /home/skroman/esp/esp-idf/components/protocomm /home/skroman/esp/esp-idf/components/pthread
 /home/skroman/esp/esp-idf/components/sdmmc /home/skroman/esp/esp-idf/components/soc /home/skroman/esp/esp-idf/components/spi_flash /home/skro
man/esp/esp-idf/components/spiffs /home/skroman/esp/esp-idf/components/tcp_transport /home/skroman/esp/esp-idf/components/tcpip_adapter /home/
skroman/esp/esp-idf/components/tinyusb /home/skroman/esp/esp-idf/amol_codes/SKSL_Common/components/touchControlBoard /home/skroman/esp/esp-idf
/components/ulp /home/skroman/esp/esp-idf/components/unity /home/skroman/esp/esp-idf/components/usb /home/skroman/esp/esp-idf/components/vfs /
home/skroman/esp/esp-idf/components/wear_levelling /home/skroman/esp/esp-idf/components/wifi_provisioning /home/skroman/esp/esp-idf/components
/wpa_supplicant /home/skroman/esp/esp-idf/components/xtensa
-- Configuring done
-- Generating done
-- Build files have been written to: /home/skroman/esp/esp-idf/amol_codes/SKSL_Common/build
[9/1029] Generating ../../partition_table/partition-table.bin
Partition table binary generated. Contents:
*******************************************************************************
# ESP-IDF Partition Table
# Name, Type, SubType, Offset, Size, Flags
nvs,data,nvs,0x9000,128K,
otadata,data,ota,0x29000,8K,
phy_init,data,phy,0x2b000,8K,
ota_0,app,ota_0,0x30000,1536K,
ota_1,app,ota_1,0x1b0000,1536K,
MyNvs,data,nvs,0x330000,128K,
storage,data,spiffs,0x350000,64K,
*******************************************************************************
[30/1029] Building C object esp-idf/mbedtls/mbedtls/library/CMakeFiles/mbedcrypto.dir/camellia.c.obj
```

- And thus finally the code is programmed in the device, it can be verified by connecting through the **Skroman Slide app**, from where we can operate the device.
- These all details of the device is even emailed to the mail-ID of the production department from where we can easily get all the QR codes, and as a virtual access.



Skroman Device (('27', 'SKSL_GWZU5z', 'POP_YIM8r3', '89000', 'Switch Box')) Inbox ×

**Skroman Production** <skromanproductionlogs@gmail.com>
to me

4:15 PM (17 minutes ago)