WDB Systems Pvt Ltd
#23, Srinilayam, 7th Cross, 5th Main, GM Palya, Bangalore 560075 INDIA • www.wdbsystems.com

Environmental Controller Device

Architecture , Design , Protocol

TABLE OF CONTENTS

DOCUMENT CHANGE HISTORY

Date	Person	Version	Reason
21-08-2020	Venu Kosuri	0.1	Initial Release
19-10-2020	Venu Kosuri	0.2	Command protocol between APP & EC added
19-11-2020	Venu Kosuri	0.3	HW Resources added, see Appendix A
25-02-2021	Venu Kosuri	0.4	added Appendix B added Appendix C BOOT LOADER architecture details EC Application architecture details Scenarios details
19-03-2021	Venu Kosuri	0.5	updated protocol as per new configuration parameter
18-04-2021	Venu Kosuri	0.6	updated protocol as per new json fields of device info , enum value of commands as per code
03-05-2021	Venu Kosuri	0.7	added SMTP info for email functionality, MQTT command acknowledge added.
03-05-2021	Venu Kosuri	0.8	added CO2 protocol params
14-09-2021	Venu Kosuri	0.9	provisioning note added section 4.2.1 update
28-10-2021	Venu Kosuri	0.10	4.9.Appendix C update , 4.9.Appendix B update 4.2.1 update for JSON element "time_zone": Section 3 update
25-03-2022	Linta	0.11	4.2.3 added (erase provisioning) 4.3.6 added (erase particular profile) 4.2 updated to add new fields to wifi /mqtt to support WPA2, MQTT SSL
27-07-2023	Venu Kosuri	0.12	critical_temp_fault low_temp_state low_humidity_state low_co2_state high_temp_state high_humidity_state high_co2_state, modbus parameters added

1. INTRODUCTION

1.1. PURPOSE AND SCOPE

This document mentions all details of architecture, design & protocol details of environmental controller system.

1.2. OPEN POINTS

- None

1.3. ASSUMPTIONS

- None

1.4. DEFINITIONS, ACRONYMS & ABBREVIATIONS

Definitions:

Component :- Software module having designated functionality

Acronyms:

Dir : - Directory

Firmware: - Software running EC Hardware

Mobile APP: - Mobile Application communicating with EC

Abbreviations:

EC :-Environmental Controller

HW : - Hardware

I/O :- Input /Output

SW : - Software

1.5. REFERENCES

[EC_product_spec_Vx (Scott)] Specification Document

2. TECHNICAL SPECIFICATIONS

This section describes high level specifications of the environmental controller system for quick reference. Pl see the document [EC_product_spec_Vx] for detailed requirements .

2.1. SYSTEM

EC is environmental controlling device used in green house environments. The different crops required different environmental conditions such as temperature, humidity, light , fertilisation, and pesticide cycles to be maintained to give good yield. EC is programmable device to maintain such environmental conditions for different crops.

The environmental controller at the simplest consists of the 6 device outlet relays plus sensors to monitor temperature , humidity & Co2 .

The basic premise of the controller is a 24 hour recycling timer- The controller will operate according to light and dark cycle parameters in a 24 hour cycle, then start again on a new 24 hour cycle. One 'profile' consists of all the desired user parameters for operating the 6 device outlet relays plus sensors.

The following are top level requirements.

- EC controls six kinds of device outlets to control the environment parameters.
- EC communicates with a mobile application over the internet using wireless LAN technology. The green house is expected to have internet connectivity using the Wifi .
- For different crops there are different environmental profiles. These profile status & configurability is done by Mobile APP. Using this APP, the current status of profiles & their parameters can be read.
- During grow cycle these profiles can be configurable & applied to the grow cycle any time using the APP.
- EC is always powered using live power.

3. ARCHITECTURE

EC Firmware is split into two independent entities running separately, loaded at different locations. Pl refer Appendix C for memory organisation.

- Boot Loader
- EC Application

In case of ESP32S2 HW platform, Boot loader component is provided by ESP32S2 SDK . Skip boot loader section 3.1.

3.1. BOOTLOADER

Boot Loader is the initial program that gets activated when EC is switched ON. The functionality of the boot loader is to take care of loading new controller application if needed and it releases the control to Controller Application. So it supports use case of firmware upgrade functional feature.

3.1.1. STATIC VIEW

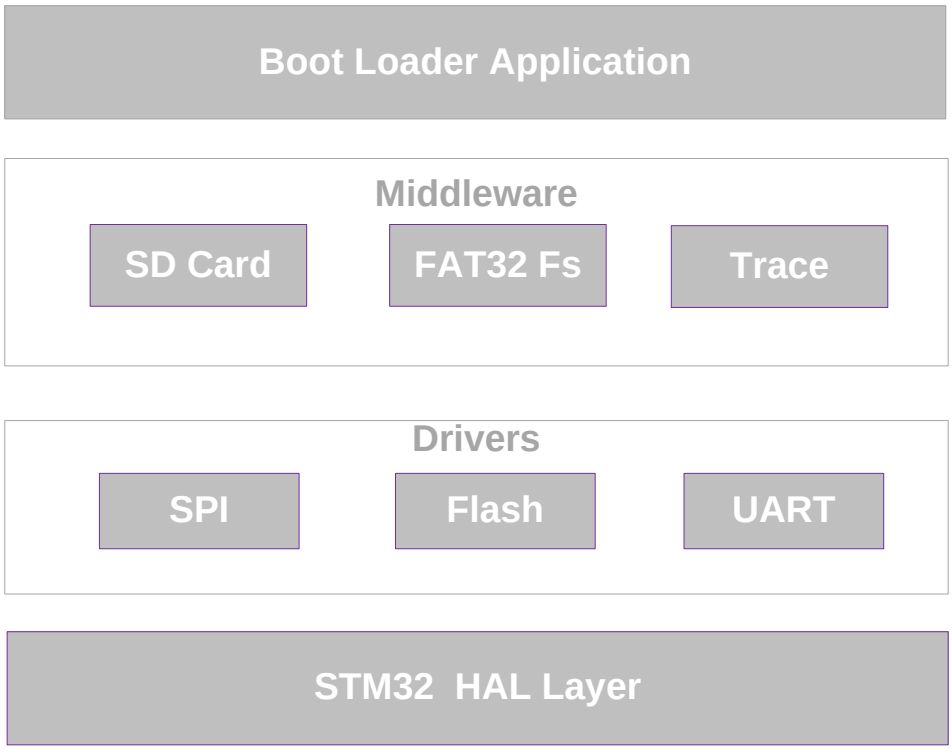

Figure 1

Boot Loader consists of the following components

- Boot Loader application
  - It checks the need for upgrading firmware & relinquishes the control to control application
- SD card

    - It consists of functionality of reading /writing at file /directory level to/from SD card
- FAT32 File System
    - It is FAT32 file system used to access SD card
    - It is open source code module i.e fat_io_lib
- Trace module
    - It has frame work for debug prints , print levels such as DBG , INFO WARN , ERROR etc.. can be controlled
- SPI driver
    - It is to access SD card physically
- FLASH driver
    - It is to access ARM Program Flash for read /write/erase
- UART driver
    - Debug Prints are sent over UART interface
- STM32 HAL Layer
    - Start up code
    - HAL layer to access HW registers , ISRs , drivers
    - Std C libraries

3.1.2. DYNAMIC VIEW

Boot Loader doesn't have any RTOS or specific scheduler as it implements only one dedicated feature i.e. firmware upgrade. The boot loader works as per the given flow chart. Pl note Boot loader check whether there is EC Application binary present or not in SD card. If it is present, it copies EC application from SD card to program FLASH

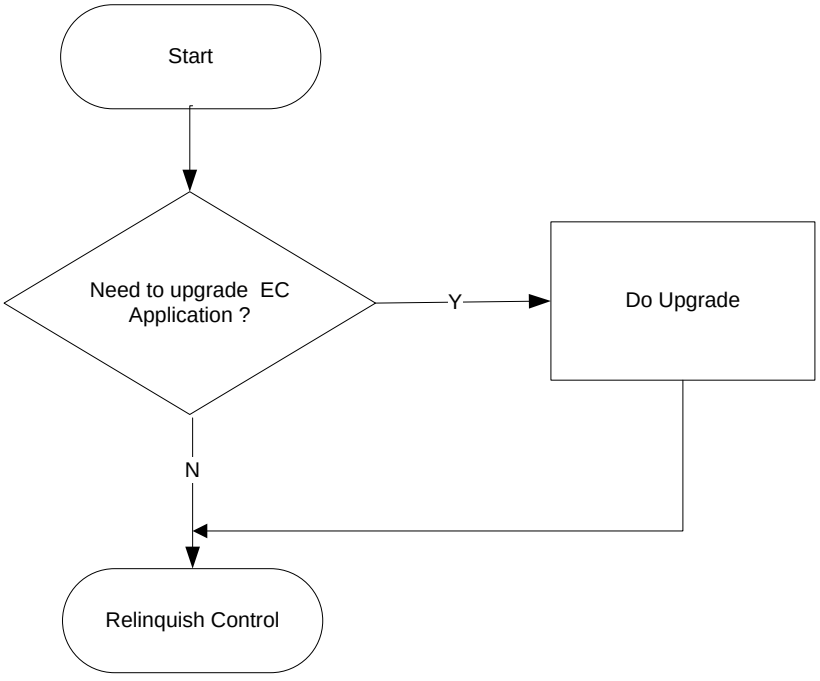

Figure 2

3.2. EC APPLICATION

This is the actual application implementing all functionality

3.2.1. STATIC VIEW

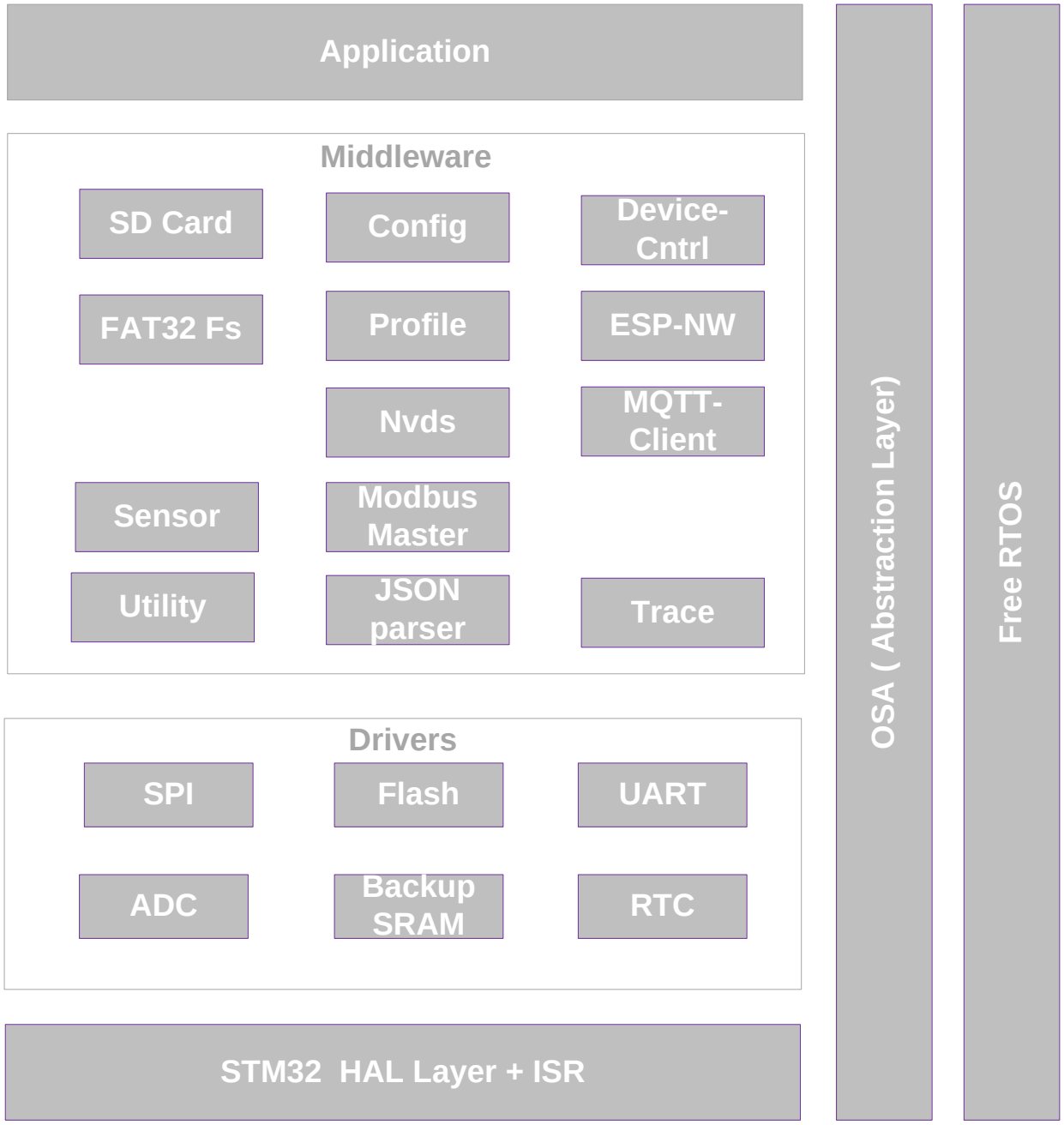

Figure 3

EC Application consists of the following components

- Application
  - It is main control application
- SD card
  - It consists of functionality of reading /writing at file /directory level to/from SD card

- FAT32 File System
  - It is FAT32 file system used to access SD card
  - It is open source code module i.e fat_io_lib
- Trace module
  - It has frame work for debug prints , print levels such as DBG , INFO WARN , ERROR etc.. can be controlled
- Device Cntrl
  - This module takes care of functionality of all device outlets as per device operation states.
- ESP NW
  - It takes care of communication functionality with ESP module to send /receive over Wifi in case of STM32 platform
  - In case of ESP32S2 , it used WIFI API provided by ESP32S2 SDK
- MQTT client
  - MQTT client responsible for MQTT protocol with remote server
  - It uses Paho MQTT open source code in case of STM32 platform .
  - It uses ESP32S2 SDK provided MQTT client
- Modbus Master
  - Implements modbus master protocol
- Sensor
  - To get sensor values such as temp, humidity , Co2 etc ...
- Config + Profile + Nvds
  - To set/get/ user/factory configuration profiles as well as device provisioning details
- FreeRTOS + OSA
  - All FreeRTOS Kernel
  - OSA abstracts all freeRTOS calls.
- SPI driver
  - It is to access SD card physically
- FLASH driver
  - It is to access ARM Program Flash for read /write/erase
- UART driver
  - Debug Prints are sent over UART interface. It is available in STM32 HW only.
- ADC driver
  - To check the Battery voltage
- RTC driver
  - To implement real time clock functionality .
- Backup Sram
  - To read/write to Back up SRAM
- STM32 HAL Layer + ISR(this component is present only on STM32 platform)
  - Start up code
  - HAL layer to access HW registers ,drivers
  - Std C libraries
- ESP32S2 libs (ESP32S2 provided libraries .)

3.2.2. DYNAMIC VIEW

EC Application uses freeRTOS Kernel along STM32 HAL layer plus HAL ISR frame work. The Application is set up into set of Tasks (or Threads). The following are the tasks in the descending order of priorities.

- Free RTOS Timer Task
- ESP Rx Task . On ESP32S2 platform , This task is Wifi Task provided by SDK.

- MQTT Task (MQTT client communicator with MQTT server using ESP). On ESP32S2 platform , This task is MQTT Connection task provided by SDK.
- Application task (Control Application task)

Free RTOS Timer Task

This is highest priority task implemented by FreeRTOS kernel. This is responsible for all timing events concerned with periodic /a periodic timers. All times are SW timers; no specific HW timers are used. At the expiry of respective SW timer, the call back is executed. The call back function is registered at the time of creating the timer.

ESP Rx Task

This takes care of communication with ESP wifi module. It process all the messages received from ESP wifi module over UART interface. This is also highest priority task.

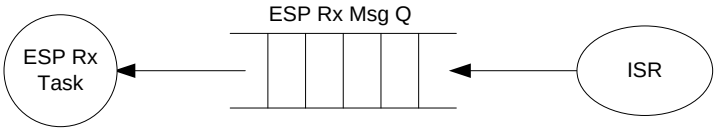

Figure 4

For ESP32S2 ignore the above description. It runs as art of wifi task provided by SDK . It takes care of functionality as Library calls of SDK.

MQTT Task

This takes care of establishing connection & communication with MQTT server.. It processes all the MQTT messages received.

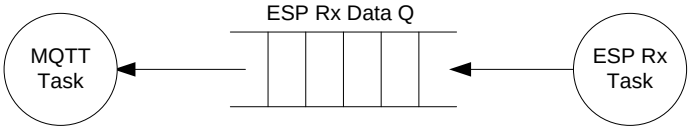

Figure 5

For ESP32S2 ignore the above description. It runs as art of MQTT task provided by SDK . It takes care of functionality as Library calls of SDK.

ApplicationTask

This takes care of functionality of EC i.e. grow cycle start/stop, provisioning, user profile configuration, device outlet control, fault detection & firmware upgrade. This is the lowest priority task.

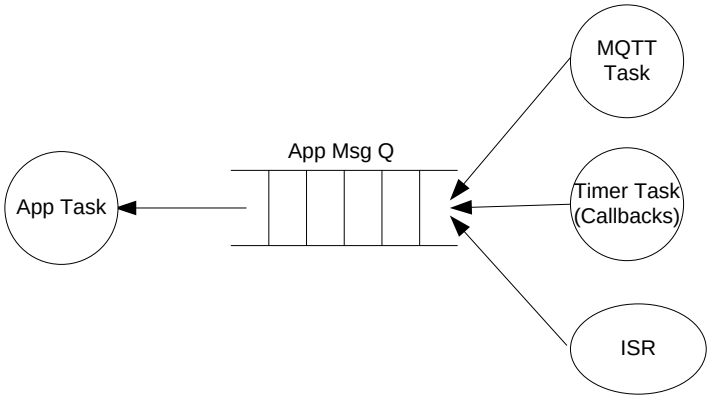

Figure 6

3.2.3. SCENARIOS

This section depicts the scenarios of EC Application. For ESP32S2 platform, In all scenarios ESP ISR is part of ESP WIFI SDK . It is shown explicitly just to ease the understanding.

3.2.3.1 EC Initialization

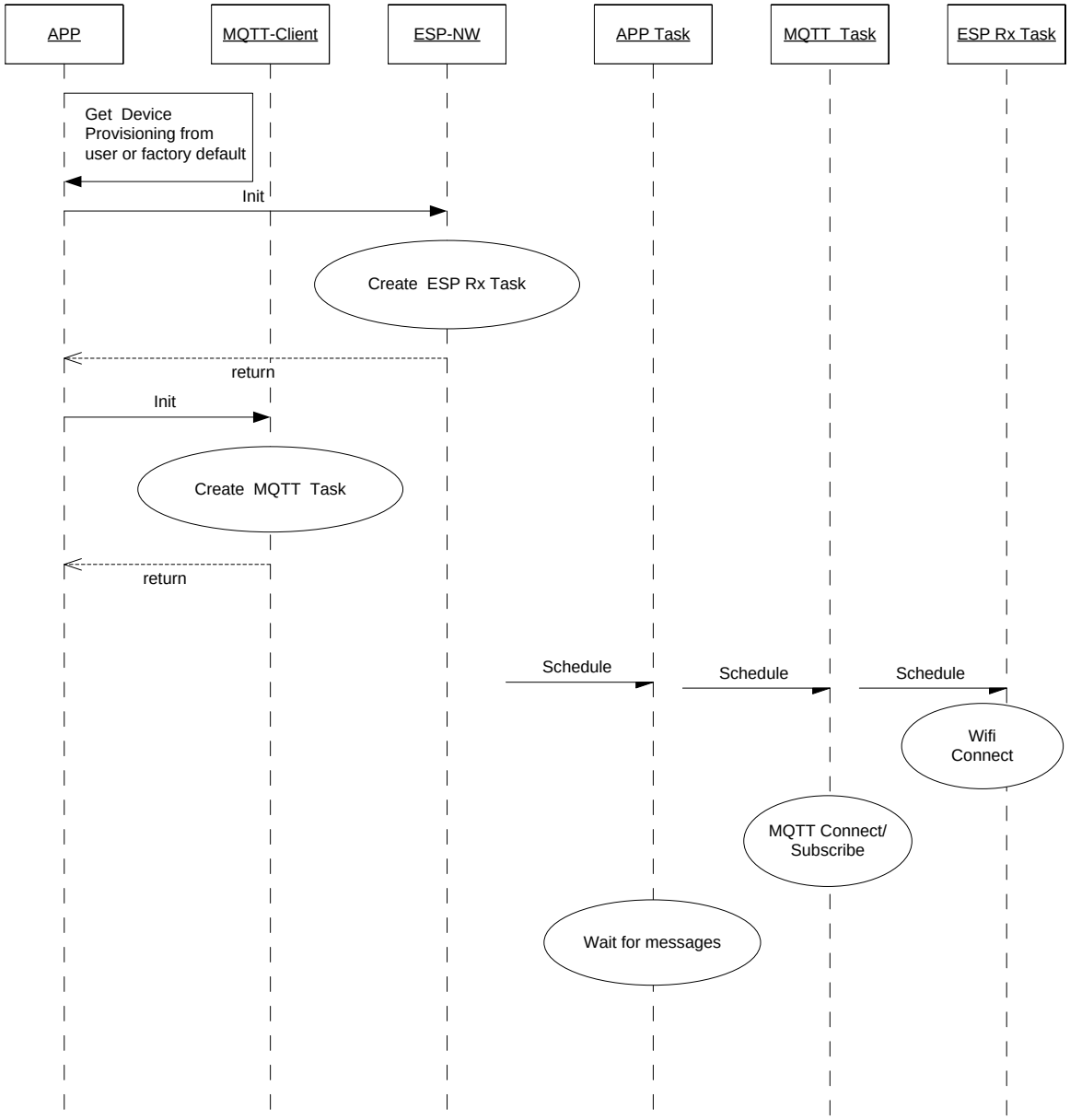

Figure 7

3.2.3.2 Grow Cycle Start

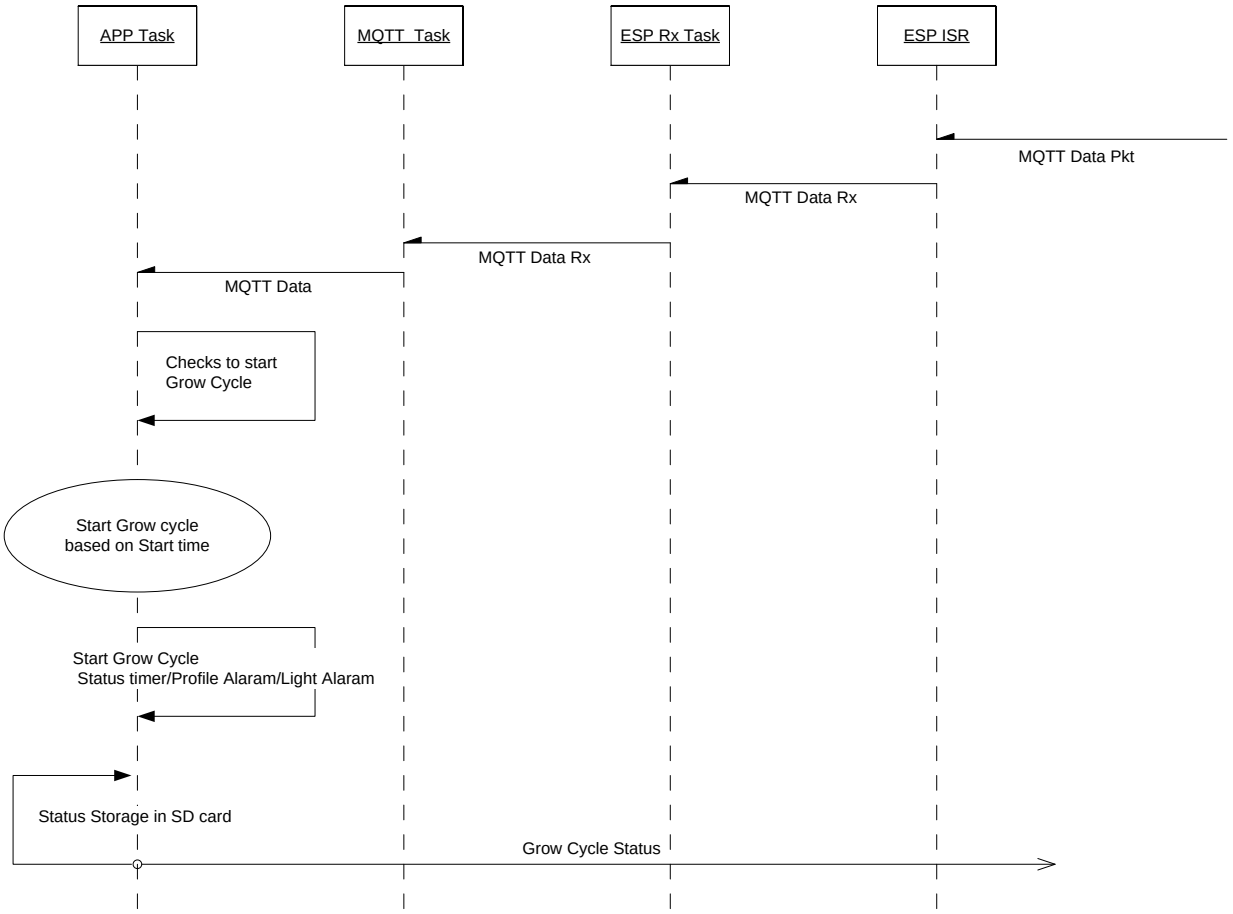

Figure 8

3.2.3.3 Grow Cycle Stop

Grow cycle stop can happen two ways either from user initiated action or designated number of days of grow cycle expired. Both cases are shown .

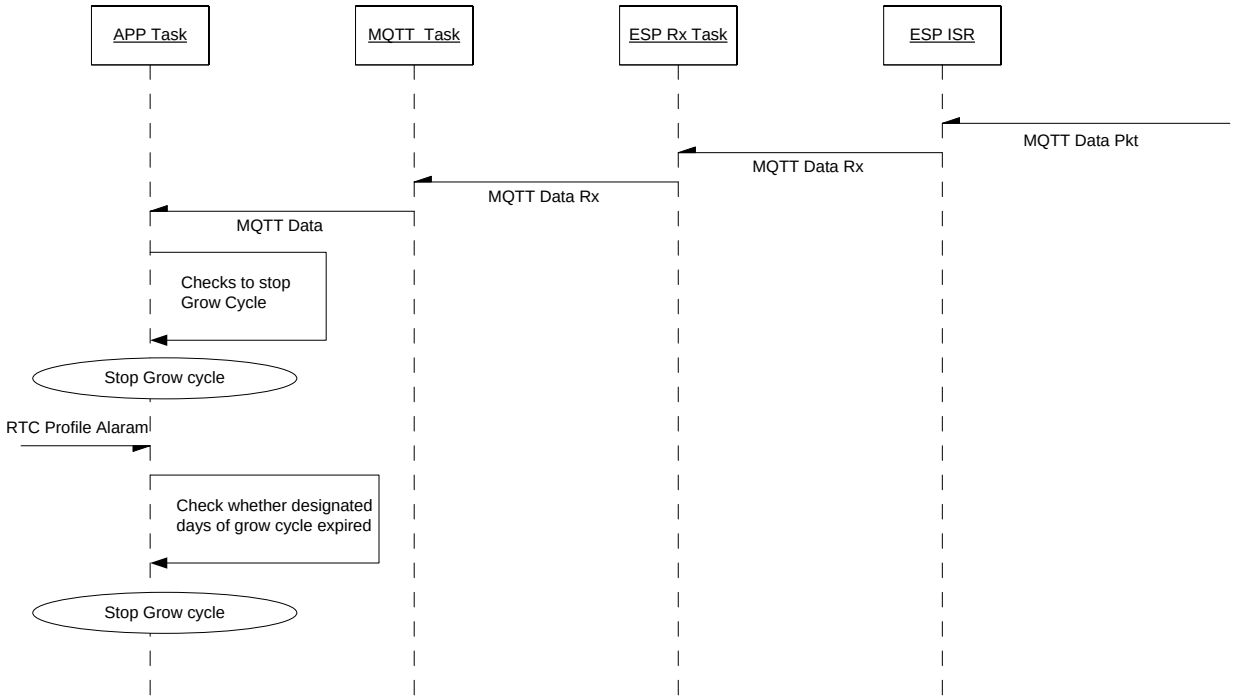

Figure 9

3.2.3.4 EC Set Provisioning

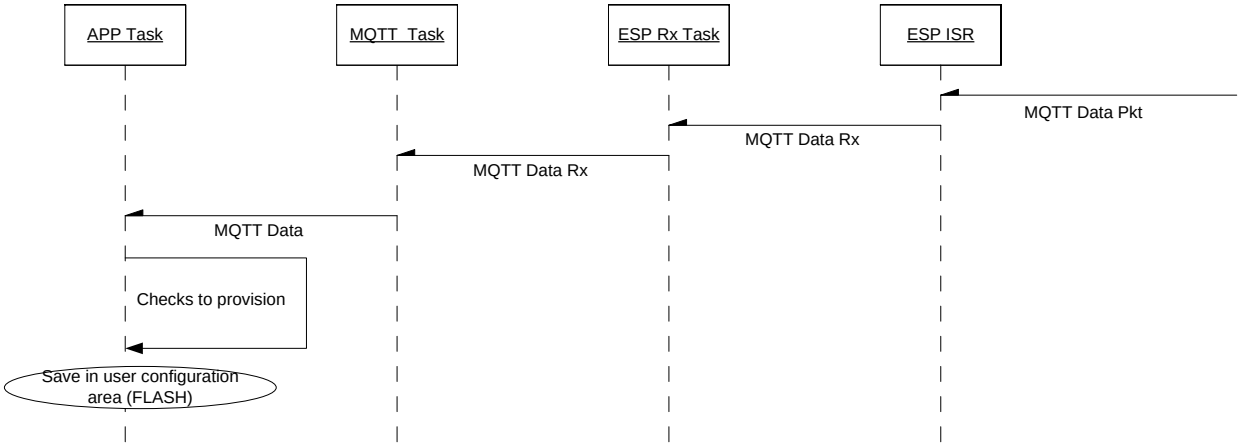

Figure 10

3.2.3.5 EC Get Provisioning

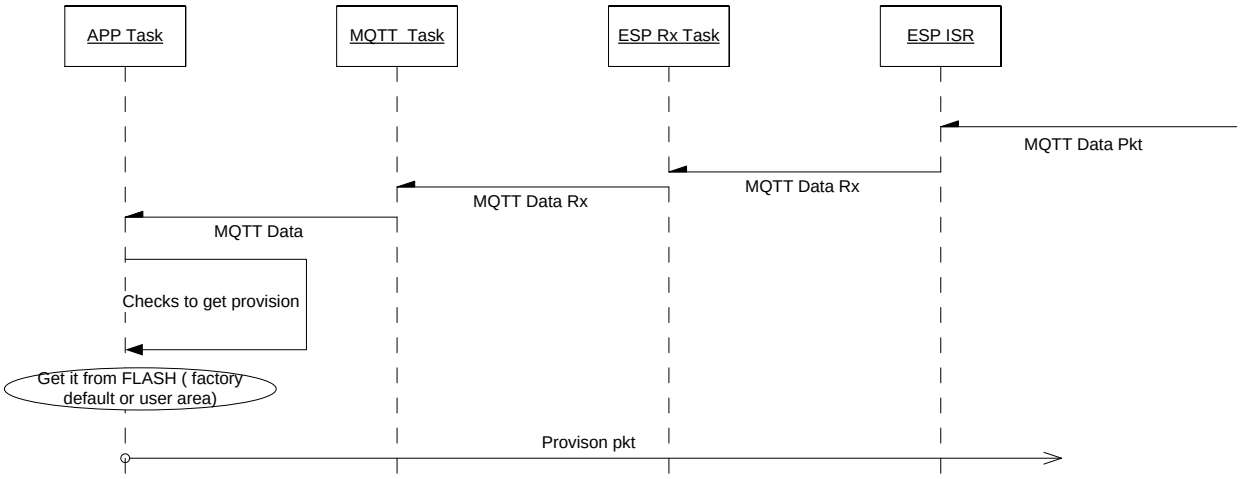

Figure 11

3.2.3.6 Profile Configuration (SET or APPLY)

Configuring user profile is possible either in grow cycle or non grow cycle. In grow cycle the profile configuration can be applied too. All these possible cases are shown in the diagram.

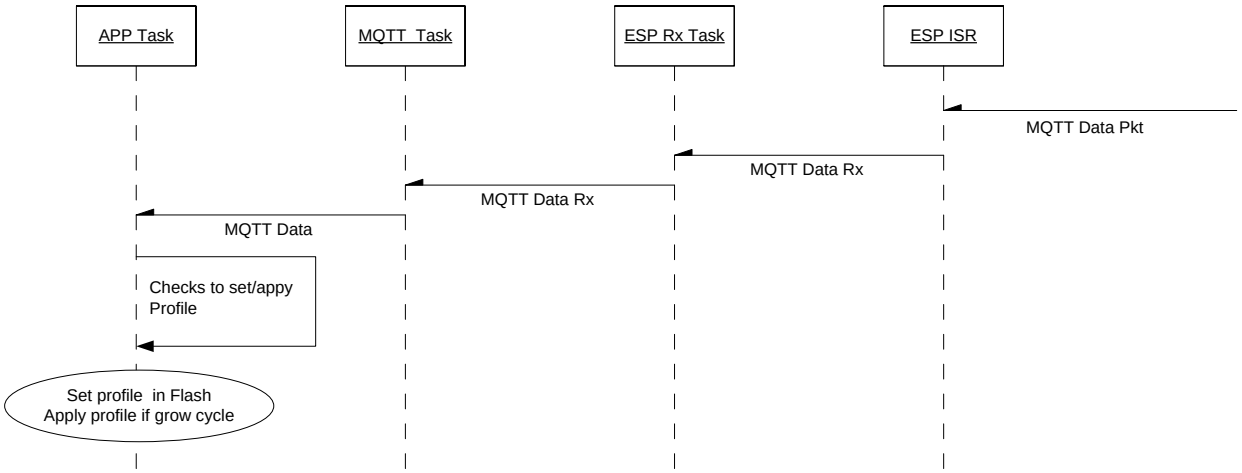

Figure 12

3.2.3.7 Get Profile

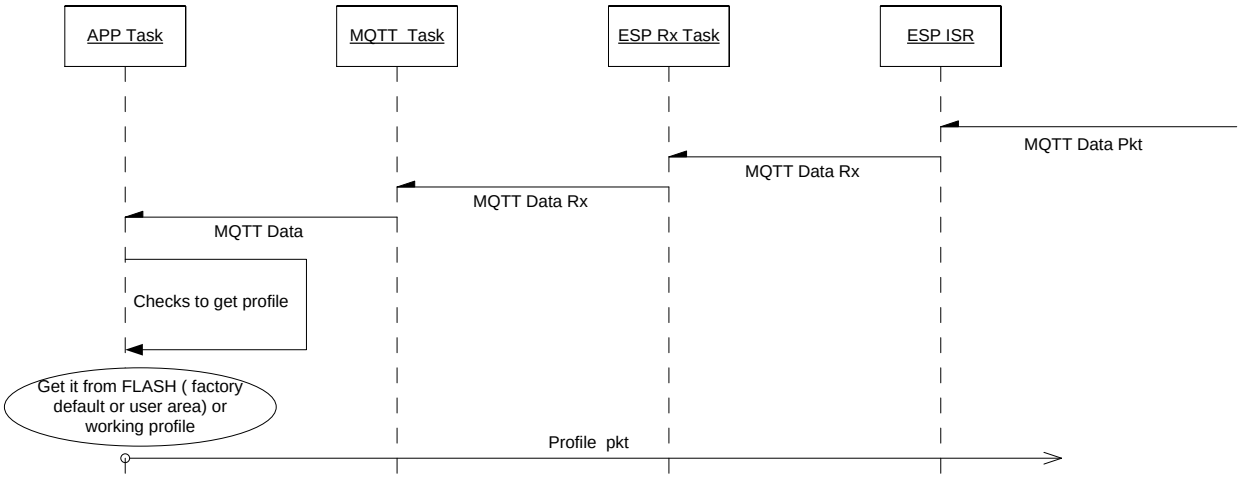

Figure 13

3.2.3.8 Faults

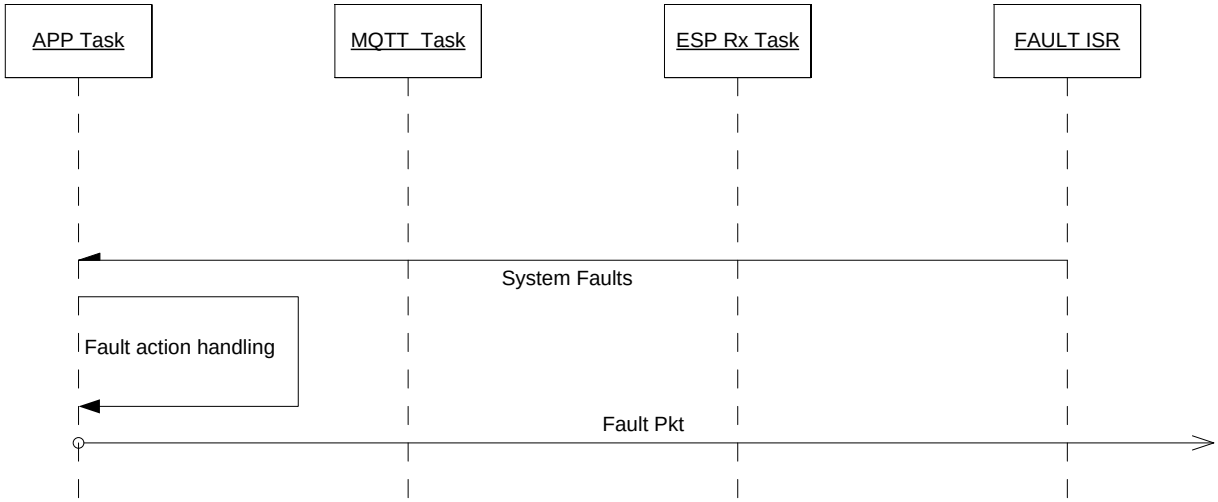

Figure 14

3.2.3.9 Firmware Upgrade

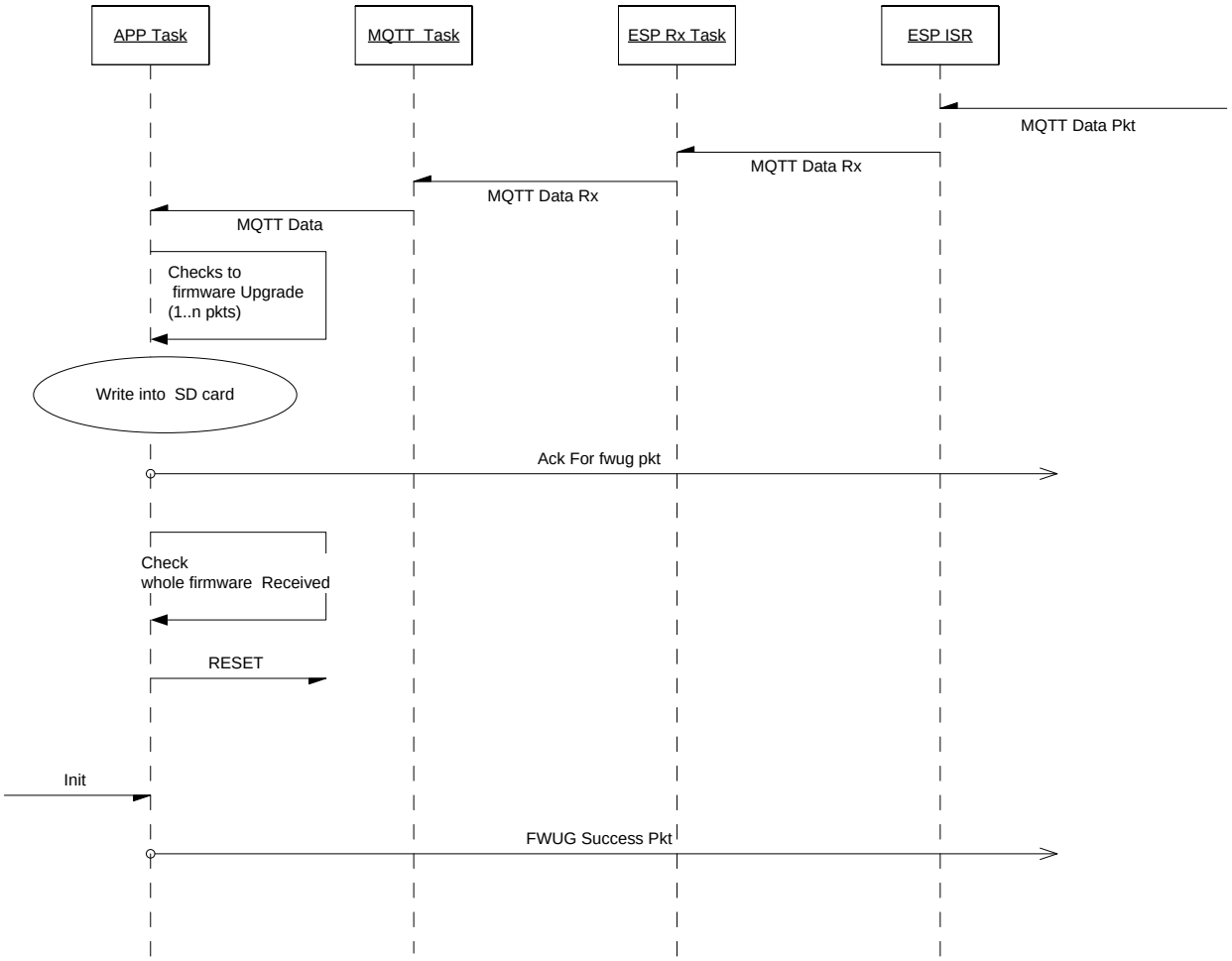
APP Task
MQTT Task
ESP Rx Task
ESP ISR
MQTT Data Pkt
MQTT Data Rx
MQTT Data Rx
MQTT Data
Checks to
firmware Upgrade
(1..n pkts)
Write into SD card
Ack For fwug pkt
Check
whole firmware Received
RESET
Init
FWUG Success Pkt

Figure 15

4. PROTOCOL

This section describes the communication protocol between the mobile APP & EC. As EC is accessible from anywhere using the mobile APP, MQTT protocol is used between mobile APP & EC using third party MQTT Broker (server) running on the cloud server.

As per MQTT protocol the communication is established using EC device id & Mobile APP id. These Ids are configured at the time of deployment. These ids help multiple ECs talking to multiple APPs using one cloud server (multiple number of green houses supported)

EC Identification:-

For each EC is identified using the following fields
device_name
device_id
farm_name
greenhouse_id
mobile_app_id

The following commands are exchanged between EC & Mobile Application

```
typedef enum APP_MQTT_CMD_TYPE
{
    APP_MQTT_PROVISION_INFO = 0,
    APP_MQTT_PROFILE_INFO,
    APP_MQTT_GROW_CYCLE_INFO,
    APP_MQTT_STATUS_INFO,
    APP_MQTT_FAULT_INFO,
    APP_MQTT_DEVICE_REQUEST,
    APP_MQTT_FW_UG_REQUEST,
    APP_MQTT_FW_UG_STATUS

}APP_MQTT_CMD_TYPE;
```

All Commands are exchanged as JSON format. The Boolean values takes 1 for true , 0 for false .

Each command reception is acknowledged by EC using following frame format.

4.1. CMD ACKNOWLEDGEMENT

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"ack":{"cmd":}
}
```

4.2. APP_MQTT_PROVISION_INFO

Purpose:- It is used to set or get the general configuration . If it is set request, the command is given by mobile APP & processed by EC. If it is get request, the command is issued as request with no fields present to EC , EC will respond as response with all the fields .

4.2.1. Set Request

Whenever provisioning is set, EC will get reset immediately to apply the latest provisioning parameters. It is not advised to do provisioning when there is grow cycle is in progress. Whenever it is required to change during grow cycle, stop the grow cycle, apply provisioning & restart the grow cycle.

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"req":{"cmd":A
PP_MQTT_GENERAL_INFO,","fields":
[{"action":"set”},
 {"wifi_info":{},},
 {"mqtt_info":{},},
 {"device_info":{},},
 {"grow_info":{},},
 {"smtp_info":{},},
]}}
```

The Respective JSON elements :-

```
Wifi_info
{
 "ssid":””,
 "passphrase":””
 “time_zone”: (-11 to 13) for STM32 board
  OR
 “time_zone”: "UTC-05:30” for ESP32S2 board, std UTC string format
 “wifi_security”:(0:PSK, 1:PEAP, 2:TLS, 3:TTLS),
 "eap_identity":””,
 "username":””,
 "password":””,
 "key_password":””,
 "ca_cert":””,
 "client_cert":””,
 "key_cert":””
}
```

Note:- All the fields may not be present . For e.g. in case of PSK, APP will send ssid, passphrase, in case of PEAP, APP will send ssid, eap_identity, username, password, in case of TLS, APP will send ssid, key_password, ca_cert, client_cert, key_cert, in case of TTLS, APP will send ssid, eap_identity, username, password.

```
mqtt_info
{
 "hostname":””,
 "portno":””,
 “mqtt_security”:(0:NO_SECURITY, 1:SSL),
 "username":””,
 "password":””,
 "mqtt_ca_cert":””,
 "mqtt_client_cert":””,
 "mqtt_key_cert":””,
 "topic":””,
 "qos_type":(0:QOS0,1:QOS1 or 2:QOS2 )
}
```

Note:- All the fields may not be present . For e.g. in case of NO_SECURITY, APP will send username, password, in case of SSL, APP will send `mqtt_ca_cert`, `mqtt_client_cert`, `mqtt_key_cert`.

```
device_info
{
 "dev_name":"",
 "dev_id":,
 "fname":"",
 "ghouse_id":,
 "app_id":
 "ver_info":
 "controller_type":

}
```

Here `"ver_info"` is EC firmware version number . This can only be read, not set by mobile APP .

```
grow_info
{
 "measure_unit": (0:ENGLISH , 1:METRIC)
 "clk":          (0:12HR, 1:24HR)
 "language":     (0:ENGLISH ,1: KOREAN,2:JAPANESE,3:CHINESE,4:GERMAN,5:
OTHERS)
 " rpt_lo_tmp":1,"lo_tmp_chk_val":18,
 "rpt_hi_tmp":1,"hi_tmp_chk_val":40,
"rpt_critical_tmp":1,"critical_tmp_chk_val":35,
 "rpt_lo_hmdy":1,"lo_hmdy_chk_val":50,
 "rpt_hi_hmdy":1,"hi_hmdy_chk_val":80,

  "rpt_lo_co2":1,"low_co2_chk_val":800,
 "rpt_hi_co2":1,"hi_co2_chk_val":1500,

  "rpt_battery_flt":1,
  "rpt_frequency_flt":1,
  "rpt_emr_fuse_flt":1,
  "rpt_sensor_flt":1,
  "rpt_rs485_fuse_flt":1,
  "light_restrike_tmr ":1,
  "light_restrike_dur":"HH:MM:SS"
  "dev_hoodvent_nt_finish_delay": "HH:MM:SS",
  "battery_voltage": ,
  "batttery_trig_voltage": ,
  "battery_hyst_voltage": ,
  "co2_pva_vent_delay_time":
  "sensor_height":
}


smtp_info
{
 "email_address":"",
 "email_password":"",
 "smtp_host_name":"",
```

```
 "smtp_port":,
 "smtp_security": 1/0
 "to_addresses":"",
 "cc_addresses":"",
 "signature":""
}
```

4.2.2. Get Request

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"req":{"cmd":A
PP_MQTT_GENERAL_INFO,","fields":
[{"action":"get","user_profile":1/0}
]}}
```

Get Response (sent by EC)

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"res":{"cmd":A
PP_MQTT_GENERAL_INFO,"fields":
[{"action":"get"},
 {"wifi_info":{},},
 {"mqtt_info":{},},
 {"device_info":{},},
 {"grow_info":{},},
 {"smtp_info":{},},
]}}
```

4.2.3. Erase User Provisioning Request

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"req":{"cmd":A
PP_MQTT_PROVISION_INFO,","fields":
[{"action":"erase"}
]}}
```

This command is used to erase user provisioning information.

4.3. APP_MQTT_PROFILE_INFO

Purpose:- It is used to set or get the profile configuration . When it is set, the command is given by mobile APP & processed by EC. When it is get, the command is issued as request with no fields present to EC, EC will respond as response with all the fields.

4.3.1. Set Request

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"req":{"cmd":A
PP_MQTT_PROFILE_INFO,","fields":
```

```
[{"action":"set","profile_no":,"user_profile":1/0,"to_save":1/0,"apply
_now":1/0,"apply_date":"DD:MM:YYYY","apply_time":"HH:MM:SS"},
{"dev_light":{},},
{"dev_hood_vent":{},},
{"dev_circ":{},},
{"dev_vent":{},},
{"dev_fxp1":{},
{"dev_fxp2":{}
]}}
```

This command is used either to store the profile, store & apply (immediately or later date/time) or apply without saving the profile in flash . This command can be used before grow cycle to save the profile. It also can be used during the grow cycle to apply dynamic profile changes with (out) saving .

"profile_no" ranges from (1...20).

If APP does not give "apply_now", "apply_date" & "apply_time" fields , it is considered not to apply profile .

If APP does not give "apply_date", it takes same day & apply grow cycle based on the starting time.

If APP does not give "apply_time", it takes current time & applies grow cycle.

If APP does not give "to_save", APP will have the profile as local RAM copy. In the event of power shut down, this profile will not be saved. It is always advised to save the profile (user recommendation).

The Respective JSON elements :-

```
dev_light
{
 "state":2(0:suspended 1:forced 2:usual),
 "light_cycle":2(0:Always on 1:Always off 2:Normal)
 "light_on_time":"HH:MM:SS"
 "light_off_time":"HH:MM:SS"
}

dev_hood_vent
{
 "state":2(0:suspended 1:forced 2:usual),
 "op_mode":0(0:Native 1:Follow 2:Flip 3:Flexi ),
 "flexi_mode":0 (0:rpttimer mode 1:Heater 2:Cooler 3:humidifier
                    4:dehumidifier 5:CO2 6:Leak Mode),
 "follow_dev":0(0:light 2:circ 3:vent 4:fxp1
                     5:fxp2),
 "nt_mode_lc":1,
 "lc_rpt_tmr_start_delay" :"HH:MM:SS",
 "lc_rpt_tmr_on_time" : "HH:MM:SS",
 "lc_rpt_tmr_off_time" :" HH:MM:SS ",
 "dc_rpt_tmr_start_delay" :"HH:MM:SS",
```

```
 "dc_rpt_tmr_on_time" :"HH:MM:SS",
 "dc_rpt_tmr_off_time" :"HH:MM:SS",
 "lc_tmp_up_val":,
 "lc_tmp_lo_val":,
 "lc_hmdy_up_val":,
 "lc_hmdy_lo_val":,
 "co2_gen_on_val":,
 "co2_gen_off_val":,
 "co2_cyl_set_val":,
  "co2_source": 0/1,(0: cylinder, 1: generator)
 "dc_tmp_up_val":,
 "dc_tmp_lo_val":,
 "dc_hmdy_up_val":,
 "dc_hmdy_lo_val":
 "lc_on":,
 "dc_on":,
 "vent_lockout":

}
```

Note:- All the fields may not be present . For e.g. in case of heater mode, APP need not send any humidity threshold values. CO2 mode need to be added

```
dev_circ
{
 "state":2(0:suspended 1:forced 2:usual),
 "op_mode":0(0:Native 1:Follow 2:Flip 3:Flexi ),
 "flexi_mode":0 (0:rpttimer mode 1:Heater 2:Cooler 3:humidifier
                     4:dehumidifier 5:CO2 6:Leak Mode),
 "follow_dev":0(0:light 1:hood vent 3:vent 4:fxp1
                      5:fxp2),
 "nt_mode_lc":1,
 "nt_mode_dc_":1,
 "nt_lc_start_delay": "HH:MM:SS",
 "nt_dc_start_delay": "HH:MM:SS",
 "lc_rpt_tmr_start_delay" :"HH:MM:SS",
 "lc_rpt_tmr_on_time" : "HH:MM:SS",
 "lc_rpt_tmr_off_time" :" HH:MM:SS ",
 "dc_rpt_tmr_start_delay" :"HH:MM:SS",
 "dc_rpt_tmr_on_time" :"HH:MM:SS",
 "dc_rpt_tmr_off_time" :"HH:MM:SS",
 "lc_tmp_up_val":,
 "lc_tmp_lo_val":,
 "lc_hmdy_up_val":,
 "lc_hmdy_lo_val":,
 "co2_gen_on_val":,
 "co2_gen_off_val":,
 "co2_cyl_set_val":,
 "co2_source": 0/1,(0: cylinder, 1: generator)
 "dc_tmp_up_val":,
 "dc_tmp_lo_val":,
 "dc_hmdy_up_val":,
 "dc_hmdy_lo_val":
 "lc_on":,
 "dc_on":,
 "vent_lockout":
```

```
}
```

Note:- All the fields may not be present, for eg in case of heater mode , APP need not send any humidity threshold values

```
dev_vent
{
 "state":2(0:suspended 1:forced 2:usual),
 "lc_rpt_tmr":,
 "lc_rpt_start_delay" :"HH:MM:SS",
 "lc_rpt_on_time" :"HH:MM:SS",
 "lc_rpt_off_time" :"HH:MM:SS",
 "lc_chk_tmp":,
 "lc_tmp_on_val":,
 "lc_tmp_off_val":,
 "lc_chk_hmdy":,
 "lc_hmdy_on_val":,
 "lc_hmdy_off_val":,
 "dc_rpt_tmr":,
 "dc_rpt_start_delay" :"HH:MM:SS",
 "dc_rpt_on_time" :"HH:MM:SS",
 "dc_rpt_off_time" :"HH:MM:SS",
 "dc_chk_tmp":,
 "dc_tmp_on_val":,
 "dc_tmp_off_val":,
 "dc_chk_hmdy":,
 "dc_hmdy_on_val":,
 "dc_hmdy_off_val":
}
```

Note:- All the fileds may not be present . for eg in case of humiidty threshold check is not required , APP need not send any humidity threshold values

```
dev_fxp1
{
 "state":2(0:suspended 1:forced 2:usual),
 "op_mode":3( 1:Follow 2:Flip 3:Flexi ),
 "flexi_mode":0 (0:rpttimer mode 1:Heater 2:Cooler 3:humidifier
                    4:dehumidifier 5:CO2 6:Leak Mode),
 "follow_dev":0(0:light 1:hood vent 2:circ 3:vent ),
 "lc_rpt_tmr_start_delay" :"HH:MM:SS",
 "lc_rpt_tmr_on_time" : "HH:MM:SS",
 "lc_rpt_tmr_off_time" :" HH:MM:SS ",
 "dc_rpt_tmr_start_delay" :"HH:MM:SS",
 "dc_rpt_tmr_on_time" :"HH:MM:SS",
 "dc_rpt_tmr_off_time" :"HH:MM:SS",
 "lc_tmp_up_val":,
 "lc_tmp_lo_val":,
 "lc_hmdy_up_val":,
 "lc_hmdy_lo_val":,
 "co2_gen_on_val":,
 "co2_gen_off_val":,
 "co2_cyl_set_val":,
  "co2_source": 0/1,(0: cylinder, 1: generator)
 "dc_tmp_up_val":,
 "dc_tmp_lo_val":,
```

```
 "dc_hmdy_up_val":,
 "dc_hmdy_lo_val":
 "lc_on":,
 "dc_on":,
 "vent_lockout":

}
```

Note: - no Native mode is there. It does not follow Fxp2 also. All the fields may not be present. For eg in case of heater mode, APP need not send any humidity threshold values.

```
dev_fxp2
{
 "state":2(0:suspended 1:forced 2:usual),
 "op_mode":3( 1:Follow 2:Flip 3:Flexi ),
 "flexi_mode":0 (0:rpttimer mode 1:Heater 2:Cooler 3:humidifier
                    4:dehumidifier 5:CO2 6:Leak Mode),
 "follow_dev":0(0:light 1:hood vent 2:circ 3:vent ),
 "lc_rpt_tmr_start_delay" :"HH:MM:SS",
 "lc_rpt_tmr_on_time" : "HH:MM:SS",
 "lc_rpt_tmr_off_time" :" HH:MM:SS ",
 "dc_rpt_tmr_start_delay" :"HH:MM:SS",
 "dc_rpt_tmr_on_time" :"HH:MM:SS",
 "dc_rpt_tmr_off_time" :"HH:MM:SS",
 "lc_tmp_up_val":,
 "lc_tmp_lo_val":,
 "lc_hmdy_up_val":,
 "lc_hmdy_lo_val":,
 "co2_gen_on_val":,
 "co2_gen_off_val":,
 "co2_cyl_set_val":,
 "co2_source": 0/1,(0: cylinder, 1: generator)
 "dc_tmp_up_val":,
 "dc_tmp_lo_val":,
 "dc_hmdy_up_val":,
 "dc_hmdy_lo_val":,
 "dc_co2_up_val":,
 "dc_co2_lo_val":
 "lc_on":,
 "dc_on":,
 "vent_lockout":

}
```

Note: - no Native mode is there. It does not follow Fxp1 also. All the fields may not be present. For eg in case of heater mode, APP need not send any humidity threshold values.

4.3.2. ApplyRequest

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name="",
"greenhouse_id="","mobile_app_id="","ver":"","flag":0,"req":{"cmd":A
PP_MQTT_PROFILE_INFO,","fields":
```

```
[{"action":"apply","profile_no":,”user_profile”:1/0,”apply_now”:1/0,"a
pply_date":”DD:MM:YYYY”,"apply_time":”HH:MM:SS”}
]}}
```

”profile_no” ranges from (1...20).

This command is used when user intends to use the stored profiles either from factory default or user defined profile set. The user might have configured this profile earlier & want to use. This command is used when the user intends to change the profile any time during the grow cycle period.

4.3.3. Get Request

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"req":{"cmd":A
PP_MQTT_PROFILE_INFO,","fields":
[{"action":"get","profile_no":,”is_user_profile”:1/0,”is_working_profi
le”:1/0}
]}}
```

”profile_no” ranges from (1...20).

If ”user_profile”, the profile is taken from user configured profile list .

If ”working_profile”, the profile is taken from the running profile . This is to address the case , user changed the configuration & but not intended to store it permanently in & wants to apply for the running cycle.

Get Response (sent by EC)

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"res":{"cmd":A
PP_MQTT_PROFILE_INFO,","fields":
[{"action":"get","profile_no”:},
 {"dev_light":{}},
 {"dev_hood_vent":{}},
 {"dev_circ":{}},
 {"dev_vent":{}},
 {"dev_fxp1":{}},
 {"dev_fxp2":{}}
]}}
```

4.3.4. ModifyRequest

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"req":{"cmd":A
PP_MQTT_PROFILE_INFO,","fields":
[{"action":"modify","profile_no":,”user_profile”:1/0,"to_save":1/0,”ap
ply_now”:1/0,"apply_date":”DD:MM:YYYY”,"apply_time":”HH:MM:SS”},
{"grow_info":{},},
```

```
{"dev_light":{},},
{"dev_hood_vent":{},},
{"dev_circ":{},},
{"dev_vent":{},},
{"dev_fxp1":{},
{"dev_fxp2":{}
]}}
```

”profile_no” ranges from (1...20).

This command is used when user intends to modify some parameters of the profile. The modified parameters can be stored or not based on the user choice. This profile can be either operational in the current grow cycle or just stored profile.

Modify Response (sent by EC)

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"res":{"cmd":A
PP_MQTT_PROFILE_INFO,","fields":
[{"action":"modify","profile_no”:, ”user_profile”:1/0,"saved_ok”:1/0 }
]}}
```

4.3.5. Erase Request

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"req":{"cmd":A
PP_MQTT_PROFILE_INFO,","fields":
[{"action":"erase"}
]}}
```

This command is used to erase complete user configuration profiles. This includes erasing of provisioning information also. This is useful feature when the user wants to reconfigure EC with new settings.

4.3.6. Erase Individual Profile Request

```
{"type":"mqtt","device_name":,      device_id="",      "farm_name"="",
"greenhouse_id"="","mobile_app_id"="","ver":"","flag":0,"req":{"cmd":A
PP_MQTT_PROFILE_INFO,","fields":
[{"action":"profile_erase","profile_no”:}
]}}
```

This command is used to erase individual user configuration profiles.

4.4. APP_MQTT_GROW_CYCLE_INFO

Purpose:- It is to start or stop the grow cycle. The command is given by mobile APP & processed by EC

4.4.1. Start Request

```
{"type":"mqtt","device_name":,device_id="","farm_name"="","greenhouse_
id"="","mobile_app_id"="","ver":"1.0.0","flag":0,"req":{"cmd":APP_MQTT
_GROW_CYCLE_INFO,"fields":[{"action":"start",”grow_name”:””,
“status_priod”:,                                          "profile_no":,
“user_profile”:1/0,"start_date":”DD/MM/YYYY”,"start_time":”HH:MM:SS”,
“total_days”: }]}}
```

”profile_no” ranges from (1...20).

If APP does not give “start_date” & “start_time” fields , it is considered to start the grow cycle immediately .

If APP does not give “start_date”, it takes same day & apply grow cycle based on the starting time.

If APP does not give “start_time”, it takes current time & applies grow cycle.

4.4.2. Stop Request

```
{"type":"mqtt","device_name":,device_id="","farm_name"="","greenhouse_
id"="","mobile_app_id"="","ver":"1.0.0","flag":0,"req":{"cmd":APP_MQTT
_GROW_CYCLE_INFO,"fields":[{"action":"stop”}]}}
```

4.5. APP_MQTT_STATUS_INFO

Purpose:- It is used to send status information . The status is sent periodically with status period set by the grow cycle info.

```
{"type":"mqtt","device_name":,device_id="","farm_name"="","greenhouse_
id"="","mobile_app_id"="","ver":"","flag":0,"res":{"cmd":APP_MQTT_STAT
US_INFO,"fields":
[{"grow_info":{}},
 {"light_cycle":{}},
 {"dev_config":{}},
 {"dev_status":{}},
 {"dev_cnt":{}},
 {"sensor_data":{}},
 {"faults":{}}
]}}
```

The Respective JSON elements :-

```
grow_info
{
 "grow_name":"",
 “total_days”:,
 “days_elapsed”: ,
 "profile_no":,
```

```
 “user_profile”:0(0:factory profile , 1 user profile)
 “transit_profile_no” :0( 0:no transition , x:apply profile no )
 "day_no":,
 "week_no":,
 "day_cur_week":,
 "grow_time":"HH:MM:SS",
 "rtc_time":"DD/MM/YYYY HH:MM:SS"
}

light_cycle
{
 “cur_cycle”:0(0:dark 1:light),
 "lc_duration":"HH:MM:SS",
 "elapsed_lc_duration":"HH:MM:SS",
 "pending_lc_duration":"HH:MM:SS",
 "dc_duration":"HH:MM:SS",
 "elapsed_dc_duration":"HH:MM:SS",
 "pending_dc_duration":"HH:MM:SS",
}

dev_config
{
 "light_cfg": (0:suspended 1:forced 2:usual),
 "hood_vent_cfg":,
 "circ_cfg":,
 "vent_cfg”:,
 "fxp1_cfg:,
 "fxp2_cfg”:
}


dev_status
{
 "light_on": (1:On, 0:Off ),
 "hood_vent_on":,
 "circ_on": ,
 "vent_on": ,
 "fxp1_on": ,
 "fxp2_on":
}

dev_cnt
{
 "light_cnt":
 "hood_vent_cnt":,
 "circ_cnt": ,
 "vent_cnt": ,
 "fxp1_cnt": ,
 "fxp2_cnt":
}

sensor_data
{
 "cur_temp":,
 "cur_humidity":,
 "cur_co2":,
 "max_temp":,
```

```
 "max_humidity":,
 "max_co2":,
 "min_temp":,
 "min_humidity":,
 "min_co2":
 "low_temp_state": 1/0  TRUE/FALSE
 "high_temp_state":
 "low_humidity_state":
 "high_humidity_state":
 "low_co2_state":
 "high_co2_state":
}

faults
{
 "backup_battery_fault":(1:Yes ,0:No),
 "cur_battery_voltage":
 "ac_power_fault":,
 "hot_restrike_fault":,
 "emr_fuse_fault":,
 "rs485_fuse_fault":,
 "sensor_fault":
 "critical_temp_fault":
}
```

4.6. APP_MQTT_FAULT_INFO

Purpose:- It is used to send fault information . It is sent as & when fault occurs.

```
{"type":"mqtt","device_name":,device_id="","farm_name"="","greenhouse_
id"="","mobile_app_id"="","ver":"","flag":0,"res":{"cmd":APP_MQTT_FAUL
T_INFO,"fields":
[ {"faults":{}}
]}}

faults
{
 "backup_battery_fault":(1:Yes ,0:No),
 "ac_power_fault":,
 "hot_restrike_fault":,
 "emr_fuse_f
ault":,
 "rs485_fuse_fault":,
 "sensor_fault":
 "critical_temp_fault":
}
```

4.7. APP_MQTT_DEVICE_REQUEST

Purpose:- It is to suspend or forced or normal operation of the device outlet.

```
{"type":"mqtt","device_name":,device_id="","farm_name"="","greenhouse_
id"="","mobile_app_id"="","ver":"1.0.0","flag":0,"req":{"cmd":APP_MQTT
_DEVICE_REQUEST,
```

```
fields":[{"action":"suspend/forced/normal",”dev[“light”,”hoodvent”,”ci
rc”,”vent”,”fxp1”,”fxp2”]}]}}
```

If APP does give only devices that are to be affected, need not all devices be present. In case of all devices to be affected, all the devices will be present.

4.8. APP_MQTT_FW_UG_REQUEST

Purpose:- It is used to upgrade the firmware of EC . The command is given by mobile APP & processed by EC. This request contain series of packets with an acknowledge coming from EC

```
FW UG Pkt:-

{"type":"mqtt","device_name":,device_id="","farm_name"="","greenhouse_
id"="","mobile_app_id"="","ver":"1.0.0","flag":0,"req":{"cmd":APP_MQTT
_FW_UG_REQUEST,”fields":
[{“file_name”:””,"file_size":"",                                “seq_no“:,
“chunk_size”:,”pkt_enc_len”:,  “pkt”:} ]
}}
```

Here pkt is encoded in BASE64 format. The chunk size is taken as 512 bytes. The packet encoded length is BASE64 encoded length.

```
FW UG Pkt ACK :-

{"type":"mqtt","device_name":,device_id="","farm_name"="","greenhouse_
id"="","mobile_app_id"="","ver":"1.0.0","flag":0,"res":{"cmd":APP_MQTT
_FW_UG_REQUEST,”fields":[{“is_pkt_ok”:1/0, “seq_no“:}]}}
```

4.9. APP_MQTT_FW_UG_STATUS

Purpose:- It is used to send the status of firmware upgrade of EC . The command is given by by EC & processed by mobile APP.

```
FW UG Status Pkt:-
{"type":"mqtt","device_name":,device_id="","farm_name"="","greenhouse_
id"="","mobile_app_id"="","ver":"1.0.0","flag":0,"res":{"cmd":APP_MQTT
_FW_UG_REQUEST,”fields":[{“app_ver”:””, “grow_cycle_restored“:1/0}]}}
```

APPENDIX A : - HW RESOURCES

This section details out all STM32 HW resources used by firmware. USART3 used for printf() on Nucleo board where as USART1 is used on actual HW.

Pin	HW Block	On Nucleo	On Real HW	SW Functionality
PA0	UART4_TX	MODBUS_TX	MODBUS_TX	
PA1	UART4_RX	MODBUS_RX	MODBUS_RX	sensor MODBUS
PA2	USART2_TX	EXP32_TX	EXP32_TX	
PA3	USART2_RX	EXP32_RX	EXP32_RX	ESP32 interface
PA4	SPI1_NSS	SPI CS	SPI CS	
PA5	SP11_CSK	SPI Clk	SPI Clk	
PA6	SP11_MISO	SPI MISO	SPI MISO	
PA7	SP11_MOSI	SPI MOSI	SPI MOSI	SD Card
PA8	GPIO_Output	ESP32 RESET	ESP32 RESET	ESP32 interface
PA9	USART1_TX		DEBUG_PRINT_TX	
PA10	USART1_RX		DEBUG_PRINT_RX	Printf functionality
PA11	GPIO_EXTI11	LEAK SENSOR	LEAK SENSOR	Leak mode
PA12	GPIO_EXTI12	PANIC KEY	PANIC KEY	Panic mode
PA13	GPIO_EXTI13	AC PWR FAULT	AC PWR FAULT	
PA14	GPIO_EXTI14	EMR FUSE FAULT	EMR FUSE FAULT	
PA15	GPIO_EXTI15	FUES - RS485	FUES - RS485	
PB0	ADC1_IN8	BATT VOLT A/D	BATT VOLT A/D	Back up Battery Measurement
PB1	GPIO_Output	BATT TEST PULSE	BATT TEST PULSE	
PB2	GPIO_Output	LIGHT RELAY	LIGHT RELAY	
PB3	GPIO_Output	HV RELAY	HV RELAY	
PB4	GPIO_Output	CIRC RELAY	CIRC RELAY	
PB5	GPIO_Output	VENT RELAY	VENT RELAY	
PB6	GPIO_Output	FXP1 RELAY	FXP1 RELAY	
PB7	GPIO_Output	FXP2 RELAY	FXP2 RELAY	
PB8	GPIO_Output	LIGHT LED	LIGHT LED	
PB9	GPIO_Output	HV LED	HV LED	
PB10	GPIO_EXTI10	HOT RE-STRIKE	HOT RE-STRIKE	
PB12	GPIO_Output	CIRC LED	CIRC LED	
PB13	GPIO_Output	VENT LED	VENT LED	
PB14	GPIO_Output	FXP1 LED	FXP1 LED	
PB15	GPIO_Output	FXP2 LED	FXP2 LED	
PC13	GPIO_Output	ESP32_PROG	ESP32_PROG	ESP32 interface
PC14	GPIO_Output	SENSOR LED	SENSOR LED	
PD8	USART3_TX	DEBUG_PRINT_TX		
PD9	USART3_RX	DEBUG_PRINT_RX		Printf functionality

APPENDIX B : - HW PIN ALLOTMENT

This section details out all pins on real hardware for STM32

Pin No	Name	Type	Functionality	Designated functionality
1	VBAT	Power		
2	PC13	I/O	GPIO_Output	ESP32_PROG
3	PC14	I/O	GPIO_Output	SENSOR LED
4	PC15			
5	PH0			
6	PH1			
7	NRST	Reset		
8	VSSA			
9	VDDA			
10	PA0		UART4_TX	MODBUS_TX
11	PA1		UART4_RX	MODBUS_RX
12	PA2		USART2_TX	EXP32_TX
13	PA3		USART2_RX	EXP32_RX
14	PA4		SPI1_NSS	SPI CS
15	PA5		SP11_CSK	SPI Clk
16	PA6		SP11_MISO	SPI MISO
17	PA7		SP11_MOSI	SPI MOSI
18	PB0		ADC1_IN8	BATT VOLT A/D
19	PB1	I/O	GPIO_Output	BATT TEST PULSE
20	PB2	I/O	GPIO_Output	LIGHT RELAY
21	PB10	I/O	GPIO_EXTI10	HOT RE-STRIKE
22	VCAP			
23	VSS			
24	VDD			
25	PB12	I/O	GPIO_Output	CIRC LED
26	PB13	I/O	GPIO_Output	VENT LED
27	PB14	I/O	GPIO_Output	FXP1 LED
28	PB15	I/O	GPIO_Output	FXP2 LED
29	PA8	I/O	GPIO_Output	ESP32_RESET
30	PA9		USART1_TX	DEBUG_PRINT_TX
31	PA10		USART1_RX	DEBUG_PRINT_RX
32	PA11		GPIO_EXTI11	LEAK SENSOR
33	PA12		GPIO_EXTI12	PANIC KEY
34	PA13		GPIO_EXTI13	AC PWR FAULT
35	VSS			
36	VDD			
37	PA14		GPIO_EXTI14	EMR FUSE FAULT

38	PA15		GPIO_EXTI15	FUES - RS485
39	PB3	I/O	GPIO_Output	HV RELAY
40	PB4	I/O	GPIO_Output	CIRC RELAY
41	PB5	I/O	GPIO_Output	VENT RELAY
42	PB6	I/O	GPIO_Output	FXP1 RELAY
43	PB7	I/O	GPIO_Output	FXP2 RELAY
44	BOOT0			
45	PB8	I/O	GPIO_Output	LIGHT LED
46	PB9	I/O	GPIO_Output	HV LED
47	VSS			
48	VDD			

The given below are pins for ESP32S2 :-

Label	Pin	Type	FUNCTION
GND	1, 2, 30, 42, 43, 46-65		GND
3V3	3		PWR
IO0	4	I	BOOT_OPTION (GPIO0)
IO1	5	A/D	BATTVOLT_A/D
IO2	6	O	BATT_TEST_PULSE
IO3	7	I	HOT_RE-STRIKE
IO4	8	O	FXP2_RELAY
IO5	9	O	FXP1_RELAY
IO6	10	O	VENT_RELAY
IO7	11	O	CIRC_RELAY
IO8	12	O	HV_RELAY
IO9	13	O	LIGHT_RELAY
IO10	14	I	ACPWR_FAULT
IO11	15		NOT USED
IO12	16	UART	MODBUS_ DE_/RE
IO13	17	I	FUSE_RS485
IO14	18		PANIC_KEY
IO15	19	OSC	XTAL_32K_P
IO16	20	OSC	XTAL_32K_N
IO17	21	UART1	MODBUS_TXD
IO18	22	UART1	MODBUS_RXD
IO19	23	USB	USB_D- (NC)
IO20	24	USB	USB_D+ (NC)
IO21	25		FXP2_LED
IO26	26	O	FXP1_LED
NC	27	NA	NC
IO33	28	O	VENT_LED
IO34	29	O	SPI_CS
IO35	31	SPI	SPI2_MOSI

IO36	32	SPI	SPI2_SCK
IO37	33	SPI	SPI2_MISO
IO38	34	O	SD_CARD_DETECT
IO39	35	O	CIRC_LED
IO40	36	O	HV_LED
IO41	37	O	LIGHT_LED
IO42	38	NA	SENSOR_LED
TXD0	39	UART0	DEBUG_TXD
RXD0	40	UART0	DEBUG_RXD
IO45	41	NA	NC
IO46	44	I	NC
EN	45	I	EN

APPENDIX C : - MEMORY LAYOUT

This section details out memory organisation for different boards.

NUCLEO_F439ZI_EVK_BOARD

Total Flash Size: - 2MB
Address Range: - (0x8000000 - 0x80200000)

Boot Loader
Address Range: - (0x8000000 - 0x800FFFF)
Size:- 64KB

Factory Configuration Settings
Address Range: - (0x8010000 - 0x80200000)
Size: - 64KB

User Configuration Settings
Address Range: - (0x8020000 - 0x803FFFFF)
Size: - 64KB

EC Application
Address Range: - (0x8040000 - 0x80 0x08200000)

ESP32S2_Board

Total Flash Size: - 4MB
Address Range: - (0x000000 - 0x400000)

Boot Loader (part of ESP SDK)
Start Address : - 0x1000
Size:- 32KB

Partition Table (part of ESP SDK)
Start Address : - 0x8000
Size:- 4KB

NVS (part of ESP SDK)
Address Range: - 0x9000
Size:- 16KB

OTA Data (part of ESP SDK, used by Boot Loader , set by Application, mentions where the latest EC running)
Start Address : - 0xD000
Size:- 8KB

Phy Init (part of ESP SDK, used by Boot Loader , set by Application)
Start Address Range: - 0xF000
Size:- 8KB

EC Application (factory default)
Start Address : - 0x10000

Size:- 1MB

EC Application OTA1 (used during Firmware Upgrade)
Start Address : - 0x130000
Size:- 1MB

EC Application OTA2 (used during Firmware Upgrade)
Start Address : - 0x250000
Size:- 1MB

Factory Configuration Settings (factory profile + certificates)
Address Range: - 0x3CE000
Size: - bottom 200 KB th position from last 1MB

User Configuration Settings
Address Range: - 0x3D6000
Size: - bottom 168 KB th position from last 1MB