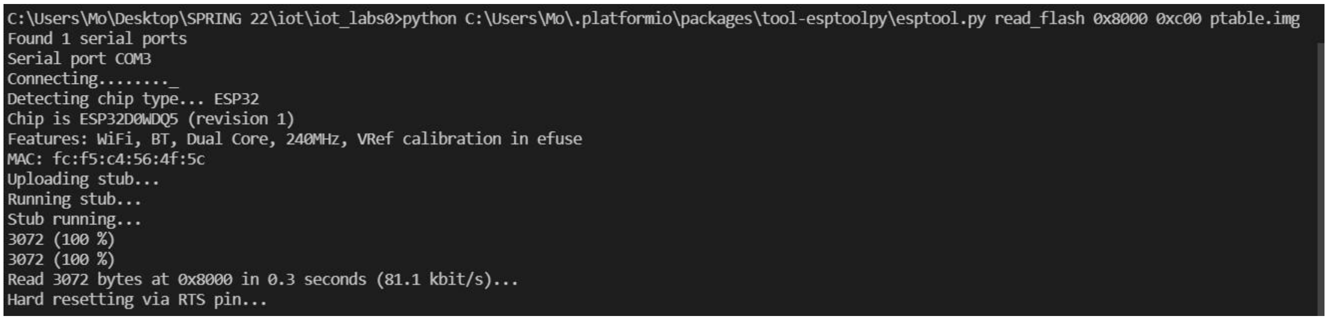
IoT Security and Privacy

Assignment 4 – ESP32 Flash Hack
(70 points)

Group Members:
Mohit Palliyil Sathyaseelan
Priyanka AbhijitTamhankar

Questions

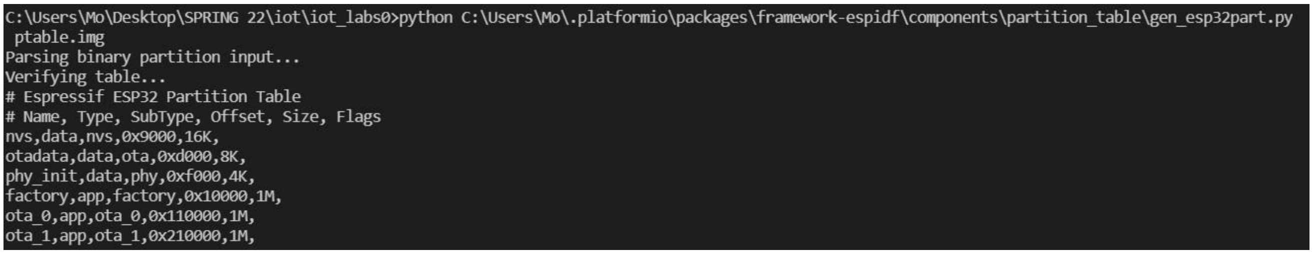
1. **Please refer to [1] for the use of *esptool*. The following command will retrieve the partition table of the IoT kit flash in the binary format:**
    a. *python C:\Users\$USER\.platformio\packages\tool-esptoolpy\esptool.py read_flash 0x8000 0xc00 ptable.img*
    where 0x8000 is the start address of the partition table and 0xc00 is the length of the partition table. $USER is the username of t
    he user that installed platformio. The binary partition table is saved in *ptable.img*. Please provide a screenshot of the command running result. [10 point]



```
C:\Users\Mo\Desktop\SPRING 22\iot\iot_labs0>python C:\Users\Mo\.platformio\packages\tool-esptoolpy\esptool.py read_flash 0x8000 0xc00 ptable.img
Found 1 serial ports
Serial port COM3
Connecting........_
Detecting chip type... ESP32
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse
MAC: fc:f5:c4:56:4f:5c
Uploading stub...
Running stub...
Stub running...
3072 (100 %)
3072 (100 %)
Read 3072 bytes at 0x8000 in 0.3 seconds (81.1 kbit/s)...
Hard resetting via RTS pin...
```

Figure 1

b. Please refer to [2] for the use of *gen_esp32part.py*. The following command will print out the partition table of our IoT kit in the CSV (comma-separated values) format. The partition table shows how the flash is partitioned.
python C:\Users\$USER\.platformio\packages\frameworkespidf\components\partition_table\gen_esp32part.py ptable.img
Please provide a screenshot of the command running result. [10 point]



```
C:\Users\Mo\Desktop\SPRING 22\iot\iot_labs0>python C:\Users\Mo\.platformio\packages\framework-espidf\components\partition_table\gen_esp32part.py
 ptable.img
Parsing binary partition input...
Verifying table...
# Espressif ESP32 Partition Table
# Name, Type, SubType, Offset, Size, Flags
nvs,data,nvs,0x9000,16K,
otadata,data,ota,0xd000,8K,
phy_init,data,phy,0xf000,4K,
factory,app,factory,0x10000,1M,
ota_0,app,ota_0,0x110000,1M,
ota_1,app,ota_1,0x210000,1M,
```

Figure 2

c. Refer to [2] and explain the partition table that is printed out. [10 point]

Name field:
It can be any meaningful name, which is not significant to the ESP32. Names longer than 16 characters will be truncated.

Type field:
Partition type is specified either as app/application (0x00) or data (0x01).

Sub-type field:
The 8-bit subtype field is specific to a given partition type. ESP-IDF currently only specifies the meaning of the subtype field for app and data partition types.

When type is app, the subtype field can be specified as factory (0x00), ota_0 (0x10) … ota_15 (0x1F) or test (0x20).

- factory (0x00) is the default app partition. The bootloader will execute the factory app unless there it sees a partition of type data/ota, in which case it reads this partition to determine which OTA image to boot.
- ota_0 (0x10) … ota_15 (0x1F) are the OTA app slots. When OTA is in use, the OTA data partition configures which app slot the bootloader should boot.
- When type is data, the subtype field can be specified as ota (0x00), phy (0x01), nvs (0x02), nvs_keys (0x04), or a range of other component-specific subtypes.
- ota (0) is the OTA data partition which stores information about the currently selected OTA app slot.
- phy (1) is for storing PHY initialisation data. This allows PHY to be configured per-device, instead of in firmware.
- nvs_keys (4) is for the NVS key partition. See Non-Volatile Storage (NVS) API for more details.
- If the partition type is any application-defined value (range 0x40-0xFE), then subtype field can be any value chosen by the application (range 0x00-0xFE).

Offset and size:
Partitions with blank offsets in the CSV file will start after the previous partition, or after the partition table in the case of the first partition. Sizes and offsets can be specified as decimal numbers, hex numbers with the prefix 0x, or size multipliers K or M (1024 and 1024*1024 bytes).

Flags:
Only one flag is currently supported, encrypted. If this field is set to encrypted, this partition will be encrypted if Flash Encryption is enabled.

Explanation –

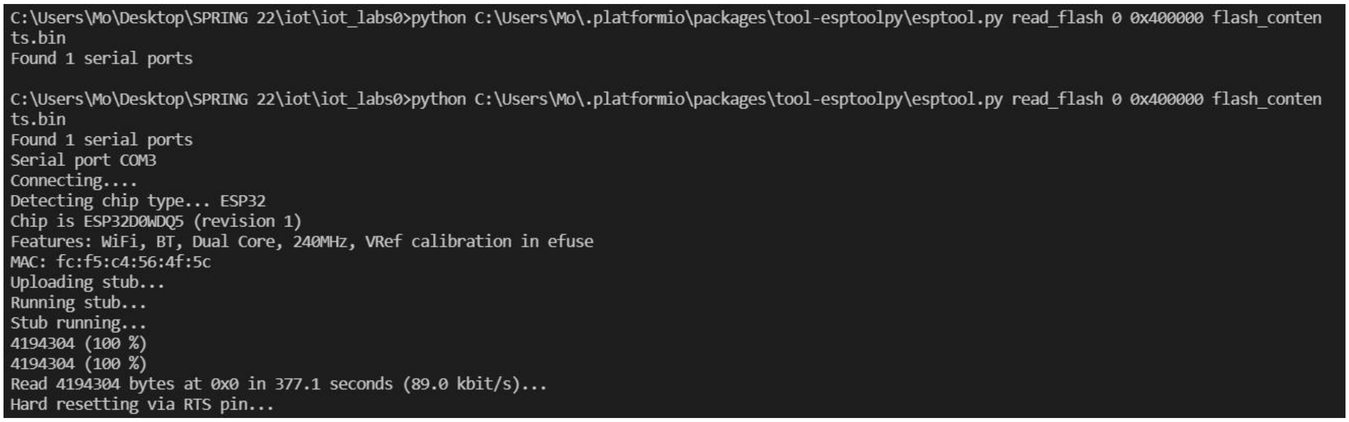
nvs,data,nvs,0x9000,16K,	This line means non-volatile storage of data type, nvs subtype with a offset of 9000 in hex and 16k bytes in size is allocated.
otadata,data,ota,0xd000,8K,	otadata holds the data for OTA updates. Bootloader consults this data to know which app to execute.
phy_init,data,phy,0xf000,4K	Used to load physical data from partition. For the remaining two lines, this is the summary printed for the “Factory app, one OTA definition” configuration.
factory,app,factory,0x10000,1M,	There are now two app partition definitions. The type of the factory app (at 0x10000) and the “OTA” app are set to “app”, but their subtypes are different.
ota_0,app,ota_0,0x110000,1	There is also a new “otadata” slot, which holds the data for OTA

M, ota_1, app, ota_1, 0x210000, 1M,	updates. The bootloader consults this data in order to know which app to execute. If “ota data” is empty, it will execute the factory app.

Table 1

Reference : https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/partition-tables.html

2. **The following command retrieves the whole flash content although the student can also refer to the partition table and print out only the occupied part of the flash.**
    a. *python C:\Users\$USER\.platformio\packages\tool-esptoolpy\esptool.py read_flash 0 0x400000 flash_contents.bin*
    where *0* is the starting address and *0x400000* is the length of the flash of the ESP32-WROOM-32 surface-mount module board that our IoT kit uses. The whole flash in the binary format is saved in *flash_contents.bin*. Please provide a screenshot of the command running result. [10 point]

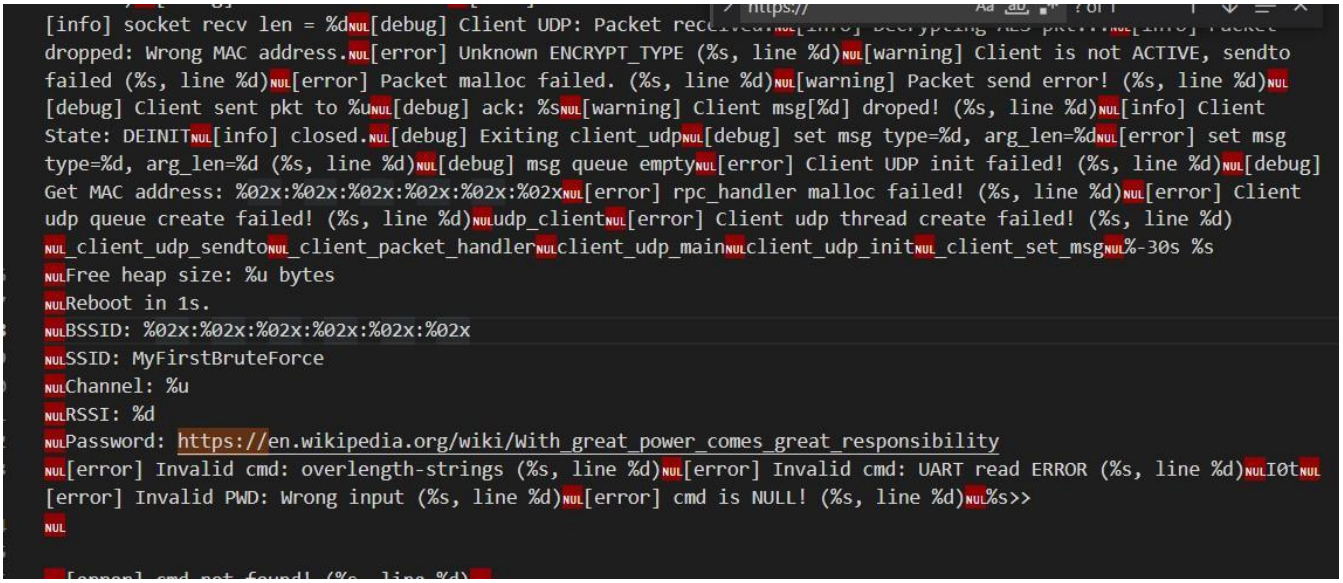


```
C:\Users\Mo\Desktop\SPRING 22\iot\iot_labs0>python C:\Users\Mo\.platformio\packages\tool-esptoolpy\esptool.py read_flash 0 0x400000 flash_conten
ts.bin
Found 1 serial ports

C:\Users\Mo\Desktop\SPRING 22\iot\iot_labs0>python C:\Users\Mo\.platformio\packages\tool-esptoolpy\esptool.py read_flash 0 0x400000 flash_conten
ts.bin
Found 1 serial ports
Serial port COM3
Connecting....
Detecting chip type... ESP32
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse
MAC: fc:f5:c4:56:4f:5c
Uploading stub...
Running stub...
Stub running...
4194304 (100 %)
4194304 (100 %)
Read 4194304 bytes at 0x0 in 377.1 seconds (89.0 kbit/s)...
Hard resetting via RTS pin...
```

Figure 3

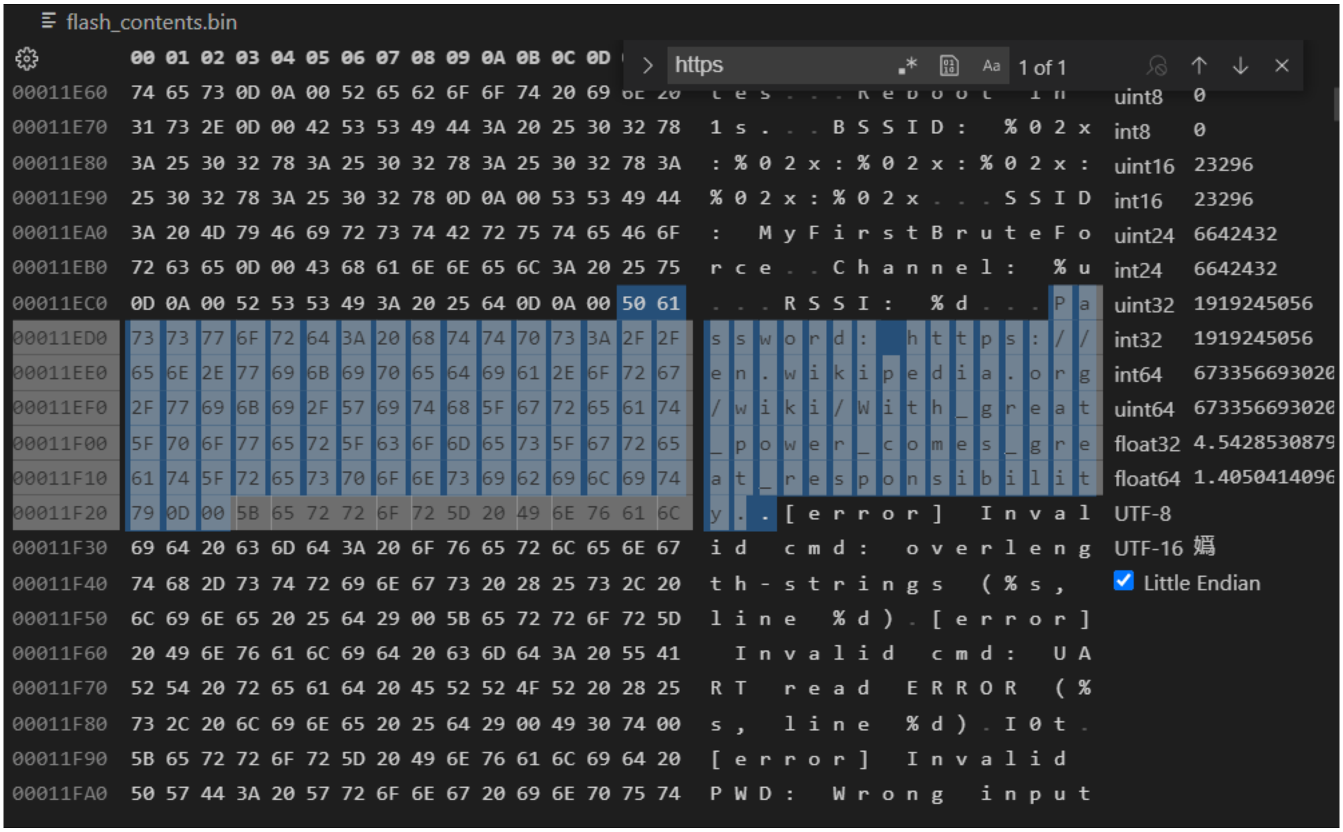
    b. Students can use a hex editor (e.g. wxhexeditor, imhex) to search the WiFi credentials in the flash dump. Please provide a screenshot of found WiFi credentials (e.g. password or key) using a hex editor. [10 point]



```
[info] socket recv len = %dNUL[debug] Client UDP: Packet received.NUL[info] Decrypting AES pkt...NUL[info] Packet
dropped: Wrong MAC address.NUL[error] Unknown ENCRYPT_TYPE (%s, line %d)NUL[warning] Client is not ACTIVE, sendto
failed (%s, line %d)NUL[error] Packet malloc failed. (%s, line %d)NUL[warning] Packet send error! (%s, line %d)NUL
[debug] Client sent pkt to %uNUL[debug] ack: %sNUL[warning] Client msg[%d] droped! (%s, line %d)NUL[info] Client
State: DEINITNUL[info] closed.NUL[debug] Exiting client_udpNUL[debug] set msg type=%d, arg_len=%dNUL[error] set msg
type=%d, arg_len=%d (%s, line %d)NUL[debug] msg queue emptyNUL[error] Client UDP init failed! (%s, line %d)NUL[debug]
Get MAC address: %02x:%02x:%02x:%02x:%02x:%02xNUL[error] rpc_handler malloc failed! (%s, line %d)NUL[error] Client
udp queue create failed! (%s, line %d)NULudp_clientNUL[error] Client udp thread create failed! (%s, line %d)
NUL_client_udp_sendtoNUL_client_packet_handlerNULclient_udp_mainNULclient_udp_initNUL_client_set_msgNUL%-30s %s
NULFree heap size: %u bytes
NULReboot in 1s.
NULBSSID: %02x:%02x:%02x:%02x:%02x:%02x
NULSSID: MyFirstBruteForce
NULChannel: %u
NULRSSI: %d
NULPassword: https://en.wikipedia.org/wiki/With_great_power_comes_great_responsibility
NUL[error] Invalid cmd: overlength-strings (%s, line %d)NUL[error] Invalid cmd: UART read ERROR (%s, line %d)NULI0tNUL
[error] Invalid PWD: Wrong input (%s, line %d)NUL[error] cmd is NULL! (%s, line %d)NUL%s>>
NUL

NUL[error] cmd not found! (%s, line %d)NUL
```

Figure 4

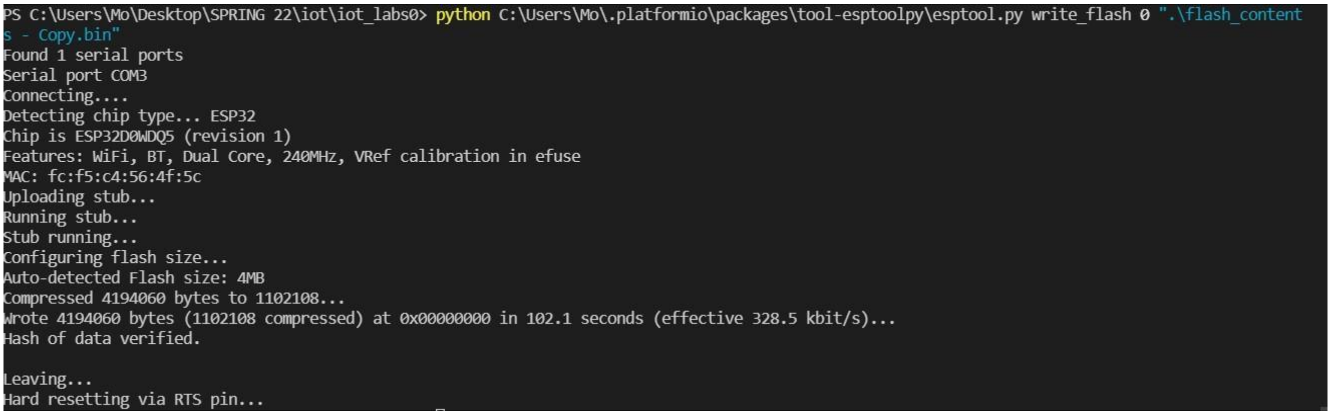


```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00011E60  74 65 73 0D 0A 00 52 65 62 6F 6F 74 20 69 6E 20  tes...Reboot in    uint8    0
00011E70  31 73 2E 0D 00 42 53 53 49 44 3A 20 25 30 32 78  1s...BSSID: %02x   int8     0
00011E80  3A 25 30 32 78 3A 25 30 32 78 3A 25 30 32 78 3A  :%02x:%02x:%02x:   uint16   23296
00011E90  25 30 32 78 3A 25 30 32 78 0D 0A 00 53 53 49 44  %02x:%02x...SSID   int16    23296
00011EA0  3A 20 4D 79 46 69 72 73 74 42 72 75 74 65 46 6F  : MyFirstBruteFo   uint24   6642432
00011EB0  72 63 65 0D 00 43 68 61 6E 6E 65 6C 3A 20 25 75  rce..Channel: %u   int24    6642432
00011EC0  0D 0A 00 52 53 53 49 3A 20 25 64 0D 0A 00 50 61  ...RSSI: %d...Pa   uint32   1919245056
00011ED0  73 73 77 6F 72 64 3A 20 68 74 74 70 73 3A 2F 2F  ssword: https://   int32    1919245056
00011EE0  65 6E 2E 77 69 6B 69 70 65 64 69 61 2E 6F 72 67  en.wikipedia.org   int64    673356693020
00011EF0  2F 77 69 6B 69 2F 57 69 74 68 5F 67 72 65 61 74  /wiki/With_great   uint64   673356693020
00011F00  5F 70 6F 77 65 72 5F 63 6F 6D 65 73 5F 67 72 65  _power_comes_gre   float32  4.5428530879
00011F10  61 74 5F 72 65 73 70 6F 6E 73 69 62 69 6C 69 74  at_responsibilit   float64  1.4050414096
00011F20  79 0D 00 5B 65 72 72 6F 72 5D 20 49 6E 76 61 6C  y..[error] Inval   UTF-8
00011F30  69 64 20 63 6D 64 3A 20 6F 76 65 72 6C 65 6E 67  id cmd: overleng   UTF-16   嫣
00011F40  74 68 2D 73 74 72 69 6E 67 73 20 28 25 73 2C 20  th-strings (%s,    ☑ Little Endian
00011F50  6C 69 6E 65 20 25 64 29 00 5B 65 72 72 6F 72 5D  line %d).[error]
00011F60  20 49 6E 76 61 6C 69 64 20 63 6D 64 3A 20 55 41   Invalid cmd: UA
00011F70  52 54 20 72 65 61 64 20 45 52 52 4F 52 20 28 25  RT read ERROR (%
00011F80  73 2C 20 6C 69 6E 65 20 25 64 29 00 49 30 74 00  s, line %d).I0t.
00011F90  5B 65 72 72 6F 72 5D 20 49 6E 76 61 6C 69 64 20  [error] Invalid 
00011FA0  50 57 44 3A 20 57 72 6F 6E 67 20 69 6E 70 75 74  PWD: Wrong input
```

Figure 5

c. Students can use a hex editor (e.g. wxhexeditor) to change the flash dump and write the changed flash dump back to the IoT kit. In an extreme case, the student may write another firmware to the IoT kit with esptool.py. Please demonstrate writing back to the IoT kit with esptool.py. Student may write either another firmware or the changed flash dump to the IoT kit. Please provide a screenshot of the command results. [20 points]
Hint: Here is a command writing the flash_contents_modified.bin to the IoT kit

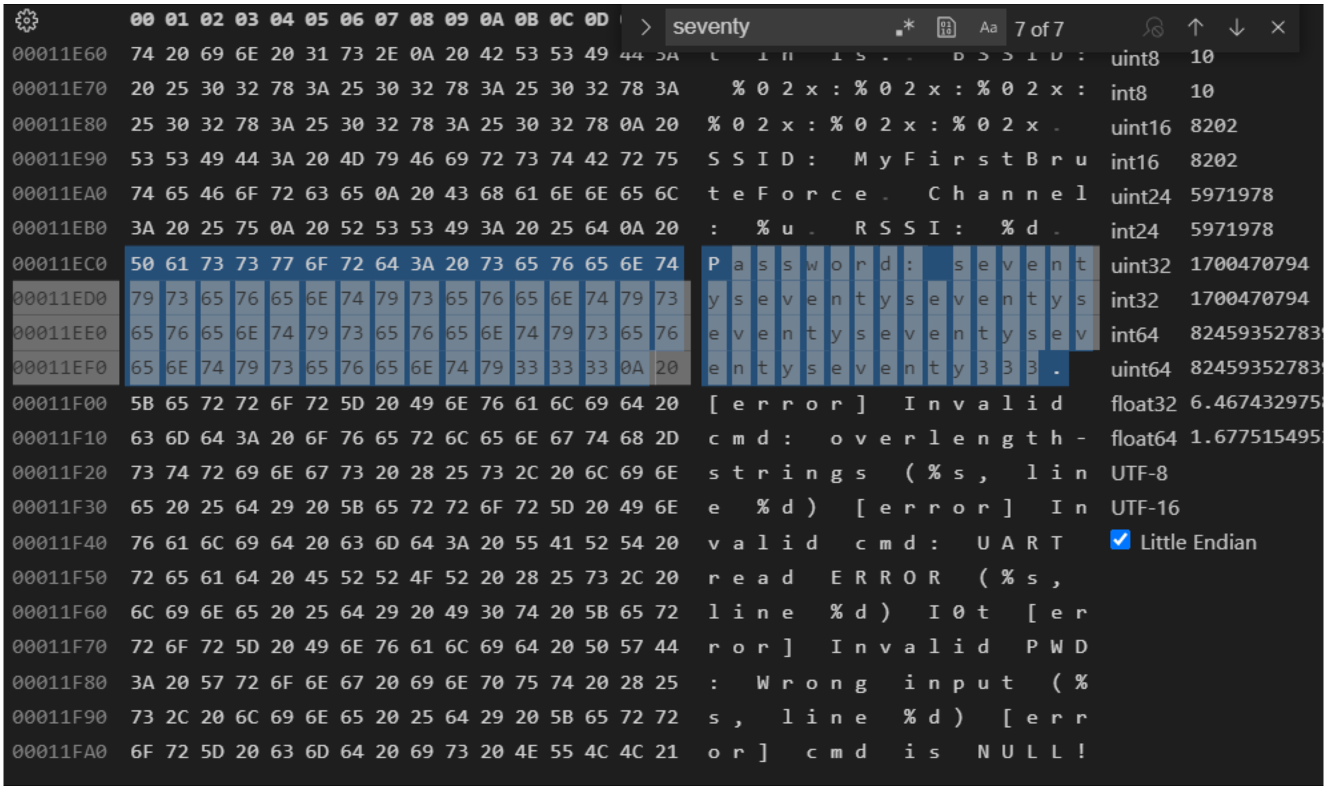
python C:\Users\$USER\.platformio\packages\tool-esptoolpy\esptool.py write_flash 0 flash_contents_modified.bin



```
PS C:\Users\Mo\Desktop\SPRING 22\iot\iot_labs0> python C:\Users\Mo\.platformio\packages\tool-esptoolpy\esptool.py write_flash 0 ".\flash_content
s - Copy.bin"
Found 1 serial ports
Serial port COM3
Connecting....
Detecting chip type... ESP32
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse
MAC: fc:f5:c4:56:4f:5c
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 4194060 bytes to 1102108...
Wrote 4194060 bytes (1102108 compressed) at 0x00000000 in 102.1 seconds (effective 328.5 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Figure 6



```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
00011E60  74 20 69 6E 20 31 73 2E 0A 20 42 53 53 49 44 3A  t in 1s.. BSSID:
00011E70  20 25 30 32 78 3A 25 30 32 78 3A 25 30 32 78 3A   %02x:%02x:%02x:
00011E80  25 30 32 78 3A 25 30 32 78 3A 25 30 32 78 0A 20  %02x:%02x:%02x.
00011E90  53 53 49 44 3A 20 4D 79 46 69 72 73 74 42 72 75  SSID: MyFirstBru
00011EA0  74 65 46 6F 72 63 65 0A 20 43 68 61 6E 6E 65 6C  teForce. Channel
00011EB0  3A 20 25 75 0A 20 52 53 53 49 3A 20 25 64 0A 20  : %u. RSSI: %d.
00011EC0  50 61 73 73 77 6F 72 64 3A 20 73 65 76 65 6E 74  Password: sevent
00011ED0  79 73 65 76 65 6E 74 79 73 65 76 65 6E 74 79 73  yseventyseventys
00011EE0  65 76 65 6E 74 79 73 65 76 65 6E 74 79 73 65 76  eventyseventysev
00011EF0  65 6E 74 79 73 65 76 65 6E 74 79 33 33 33 0A 20  entyseventy333.
00011F00  5B 65 72 72 6F 72 5D 20 49 6E 76 61 6C 69 64 20  [error] Invalid
00011F10  63 6D 64 3A 20 6F 76 65 72 6C 65 6E 67 74 68 2D  cmd: overlength-
00011F20  73 74 72 69 6E 67 73 20 28 25 73 2C 20 6C 69 6E  strings (%s, lin
00011F30  65 20 25 64 29 20 5B 65 72 72 6F 72 5D 20 49 6E  e %d) [error] In
00011F40  76 61 6C 69 64 20 63 6D 64 3A 20 55 41 52 54 20  valid cmd: UART
00011F50  72 65 61 64 20 45 52 52 4F 52 20 28 25 73 2C 20  read ERROR (%s,
00011F60  6C 69 6E 65 20 25 64 29 20 49 30 74 20 5B 65 72  line %d) I0t [er
00011F70  72 6F 72 5D 20 49 6E 76 61 6C 69 64 20 50 57 44  ror] Invalid PWD
00011F80  3A 20 57 72 6F 6E 67 20 69 6E 70 75 74 20 28 25  : Wrong input (%
00011F90  73 2C 20 6C 69 6E 65 20 25 64 29 20 5B 65 72 72  s, line %d) [err
00011FA0  6F 72 5D 20 63 6D 64 20 69 73 20 4E 55 4C 4C 21  or] cmd is NULL!
```

```
uint8    10
int8     10
uint16   8202
int16    8202
uint24   5971978
int24    5971978
uint32   1700470794
int32    1700470794
int64    82459352783
uint64   82459352783
float32  6.467432975
float64  1.677515495
UTF-8
UTF-16
☑ Little Endian
```

Figure 6

The password is changed from https://en.wikipedia.org/wiki/With_great_power_comes_great_responsibility to seventyseventyseventyseventyseventyseventyseventyseventyseventyseventy33.