IoT Engineer C Test Task report

- ✓ Programming language: C (Embedded)
- ✓ IDE: Visual Studio 2022
- ✓ Target: ESP32 Dual core WIFI and Bluetooth SoC
- ✓ SDK: ESP-IDF 4.4.1
- ✓ Hosted on: https://github.com/aroshani1985/IoT-Engineer-C-Test-Project

Summary:

IoT Engineer C Test Project (ESP32 C)

Secure communucation between two node based on request and reply

Protocol definition

1. Frame format: **[Start] [CMD_H] [CMD_L] [Payload_Len] [Payload ....] [EoF-1] [EoF-2]**
2. Start byte is 0xAA (one byte)
3. Command id is two bytes.
4. Maximum payload is 200 bytes
5. Maximum packet length is limmited to 206 bytes
6. End of frame indicators are two bytes [\r][\n]

Protocol implementation files

1. path is: main/com folder
2. encdec.c/h files are used for encryption and decription
3. pktformat.c/h files are used for packet format definition
4. pktreceive.c/h files are used for handling received packet
5. cmdhandler.c/h files are used for handling large number of commands with multiple items in responce
6. pktsend.c/h files are used for sending data to the client

Protocol test files

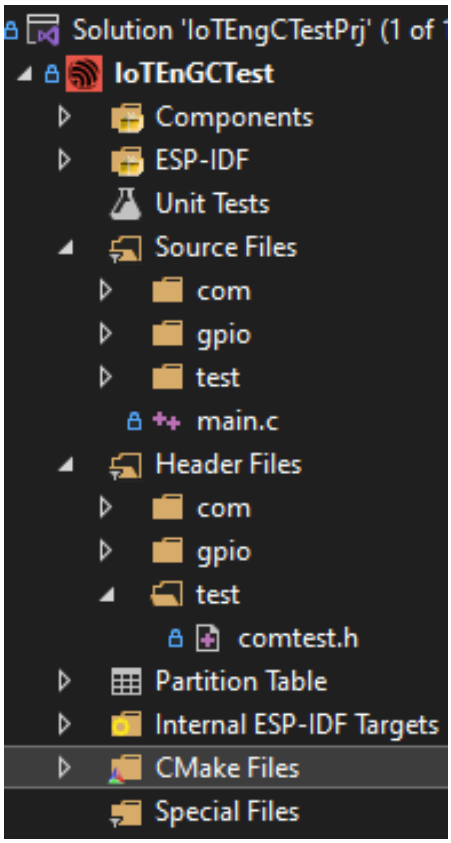
1. path is: main/test folder
2. comtest.c/h files are used for testing communication protocol implemented functions

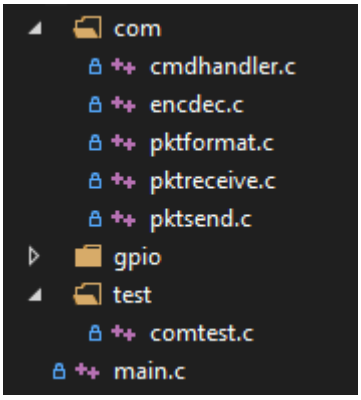
Sample output with 20 items and 3 byte for each item

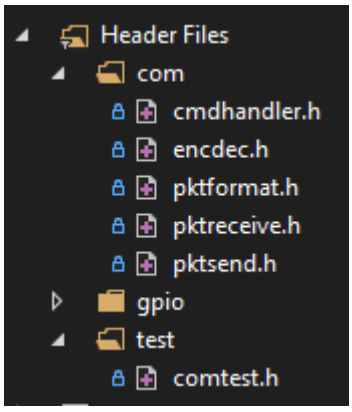
```
  Send fcn: Communication_openResponse
Send fcn: Send result: AA 00 01 00
Send fcn: Send result: A0 01 0A A0 02 0A A0 03 0A A0 04
Send fcn: Send result: 0A A0 05 0A A0 06 0A A0 07 0A A0
Send fcn: Send result: 08 0A A0 09 0A A0 0A 0A A0 0B 0A
Send fcn: Send result: A0 0C 0A A0 0D 0A A0 0E 0A A0 0F
Send fcn: Send result: 0A A0 10 0A A0 11 0A A0 12 0A A0
Send fcn: Send result: 13 0A A0 14 0A
Send fcn: Send result: 0D 0A
Send fcn: Communication_closeResponse
TestComm: CMD[0x0001], ItemCount: 20
```

Project Structure

1- **Main file**
2- **GPIO just for test the target**
3- **COM for protocol implementation**
4- **TEST for implementing test functions**







Main loop:

1- Run test function
2- Simple blink of MCU

```
Show references
void app_main(void)
{
    ////////////////////// Test encryption and decription wrapper
    //Test_01_enc_dec_fcn();
    ////////////////////////// Test packrt format functions
    //Test_02_packet_format();
    ////////////////////////////// Test function dispacher
    //Test_03_fcn_dispacher();
    ////////////////////////////// Test Communication_onDataReceived Step 1
    //Test_04_Communication_onDataReceived_s1();
    ////////////////////////////////////// Test Communication_onDataReceived Step 2
    //Test_05_Communication_onDataReceived_s2_10_items(CMD_ID_000);
    ////////////////////////////////////  Test Communication_onDataReceived Step 3
    Test_05_Communication_onDataReceived_s2_10_items(CMD_ID_001);
    ////////////////////////////////////////////////
    configure_led();
    while (1)
    {
        ESP_LOGI(TAG, "Turning the LED %s!", s_led_state == true ? "ON" : "OFF");
        blink_led(s_led_state);
        /* Toggle the LED state */
        s_led_state = !s_led_state;
        vTaskDelay(CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
    }
}
```

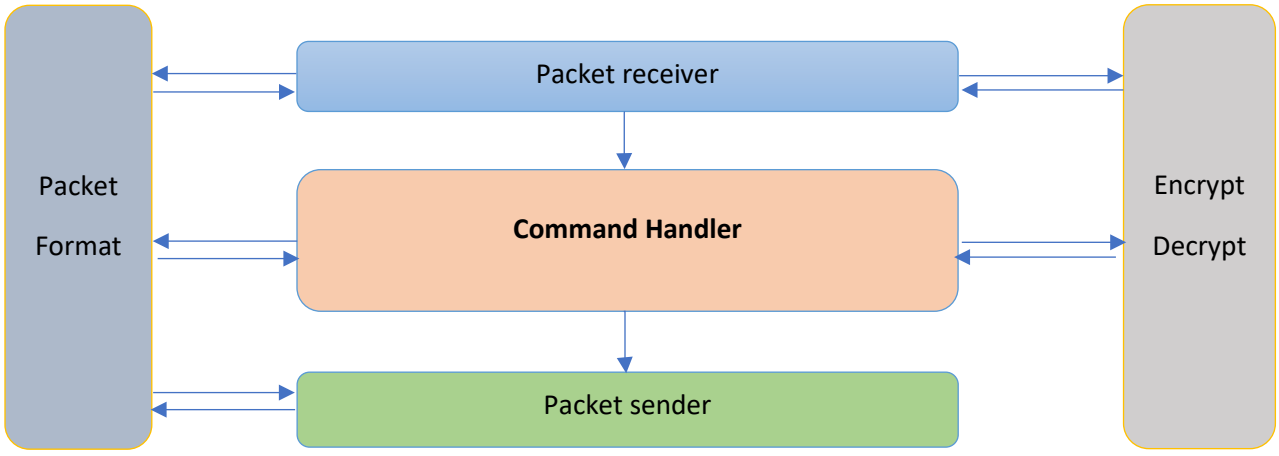
Test Functions:

```
#include <stdio.h>
#include "esp_log.h"
#include "encdec.h"
#include "pktformat.h"
#include "pktreceive.h"
#include "cmdhandler.h"


Show references
static const char *TAG_TEST_COM = "TestComm";

Show references
void Test_01_enc_dec_fcn(){ ... }

Show references
void Test_02_packet_format(){ ... }

Show references
void Test_03_fcn_dispacher(){ ... }

Show references
void Test_04_Communication_onDataReceived_s1(){ ... }

Show references
void Test_05_Communication_onDataReceived_s2_10_items(uint16_t CMD_ID){ ... }
```

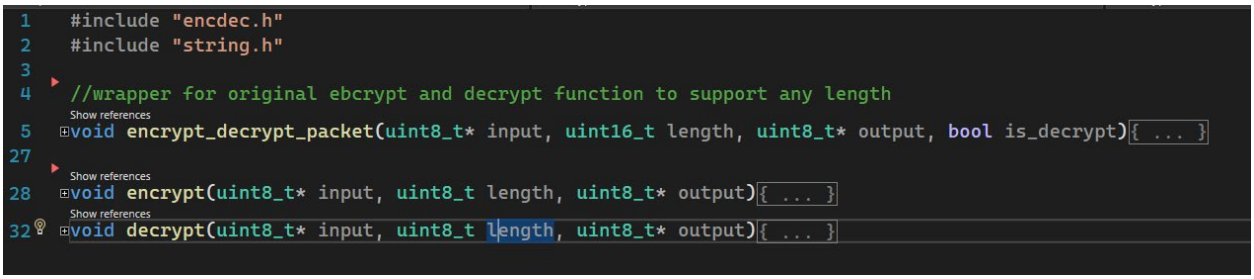
Protocol Implementation:

1- **This part is in com folder and contain 5 independent parts.**

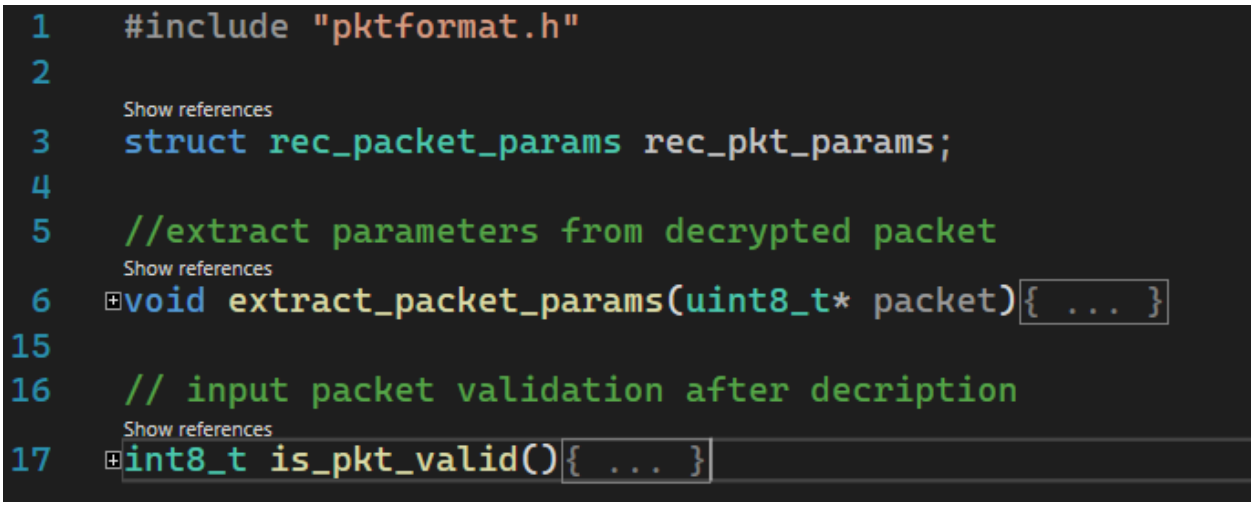
- ✓ **Encryption and decryption**
- ✓ **Packet format definition**
- ✓ **Packet receiver**
- ✓ **Packet sender**
- ✓ **Command handler and response generator**



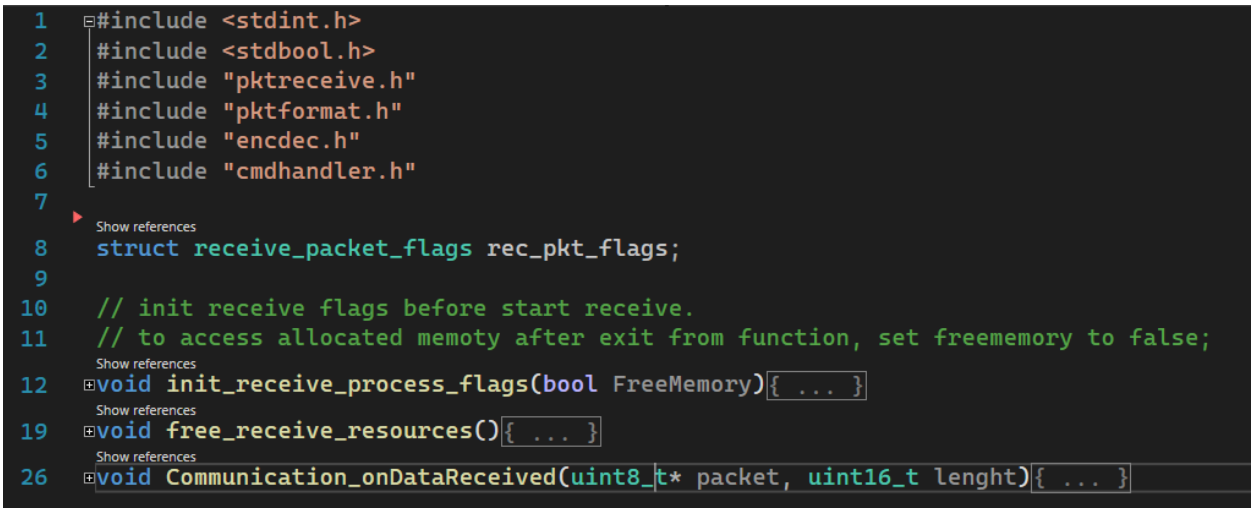
Encryption and decryption functions:



Packet format functions:



Packet receiver functions:



Packet Sender functions:

```
1   #include "pktsend.h"
2   #include "esp_log.h"
3   #include "cmdhandler.h"
4   #include "pktformat.h"
5
6   //just simple implementation for test purposes
    Show references
7   void Communication_openResponse(){ ... }
11
12  //just simple implementation for test purposes
    Show references
13  void Communication_closeResponse(){ ... }
17
18  //just simple implementation for test purposes
    Show references
19  void Communication_sendData(uint8_t* data, uint16_t length){ ... }
39
40  //just simple implementation for test purposes
    Show references
41  void Communication_appendResponse(uint8_t* data, uint16_t  length){ ... }
51
52  // send protocol header 4 byte in response packet
    Show references
53  void Communication_send_Header_Packet(uint16_t CMDIs){ ... }
65
66  // send protocol footer 2 byte
    Show references
67  void Communication_send_EndOfPacket(){ ... }
76
```

Command handler functions:

```
  1  #include "cmdhandler.h"
  2  #include "pktsend.h"
  3  #include "pktformat.h"
  4  #include "pktreceive.h"
  5  #include "encdec.h"
  6
  7  // container for each command parameters
     Show references
  8  struct command_handle_params cmd_handle_params;
  9
 10  // array of functions, each functions related to one command
     Show references
 11  cmd_process *cmd_process_array[CMD_COUNT] = {...};
 23
 24  // handle command with multiple items in response with limmted send buffer
     Show references
 25  void  CommandHandler_handle(uint16_t cmdid, uint8_t* payloadp){ ... }
 74
 75  // implementation of each command process function
     Show references
 76  void cmd_000_process(uint8_t* payloadp, uint16_t ItemIdx, uint8_t* responseDatap){ ... }
     Show references
108  void cmd_001_process(uint8_t* payloadp, uint16_t ItemIdx, uint8_t* responseDatap){ ... }
     Show references
137  void cmd_002_process(uint8_t* payloadp, uint16_t ItemIdx, uint8_t* responseDatap){ ... }
     Show references
166  void cmd_003_process(uint8_t* payloadp, uint16_t ItemIdx, uint8_t* responseDatap){ ... }
     Show references
195  void cmd_004_process(uint8_t* payloadp, uint16_t ItemIdx, uint8_t* responseDatap){ ... }
     Show references
224  void cmd_005_process(uint8_t* payloadp, uint16_t ItemIdx, uint8_t* responseDatap){ ... }
     Show references
253  void cmd_006_process(uint8_t* payloadp, uint16_t ItemIdx, uint8_t* responseDatap){ ... }
     Show references
282  void cmd_007_process(uint8_t* payloadp, uint16_t ItemIdx, uint8_t* responseDatap){ ... }
     Show references
311  void cmd_008_process(uint8_t* payloadp, uint16_t ItemIdx, uint8_t* responseDatap){ ... }
     Show references
340  void cmd_009_process(uint8_t* payloadp, uint16_t ItemIdx, uint8_t* responseDatap){ ... }
369
```