- Date: 09-05-2024
- Instruction Resource for Beginners

1. Material to Grasp the Available IoT System

- The university thesis illustrating the system:
  - ThesisVersioning0_1_0/Thesis-Template0_1_0/main.pdf at main · ngminhthanh12a3/ThesisVersioning0_1_0 (github.com)
  - ThesisVersioning0_1_0/Presentation/ThesisPresentation/slides.pdf at main · ngminhthanh12a3/ThesisVersioning0_1_0 (github.com)
- Source code of the system:
  - Server side: ngminhthanh12a3/desiot-server at 1.x.x (github.com)
  - ESP32 Gateway: ngminhthanh12a3/DESIoT_ESP32_Gateway at 1.x.x (github.com)

2. The Available IoT Architecture

3. The Available IoT Model

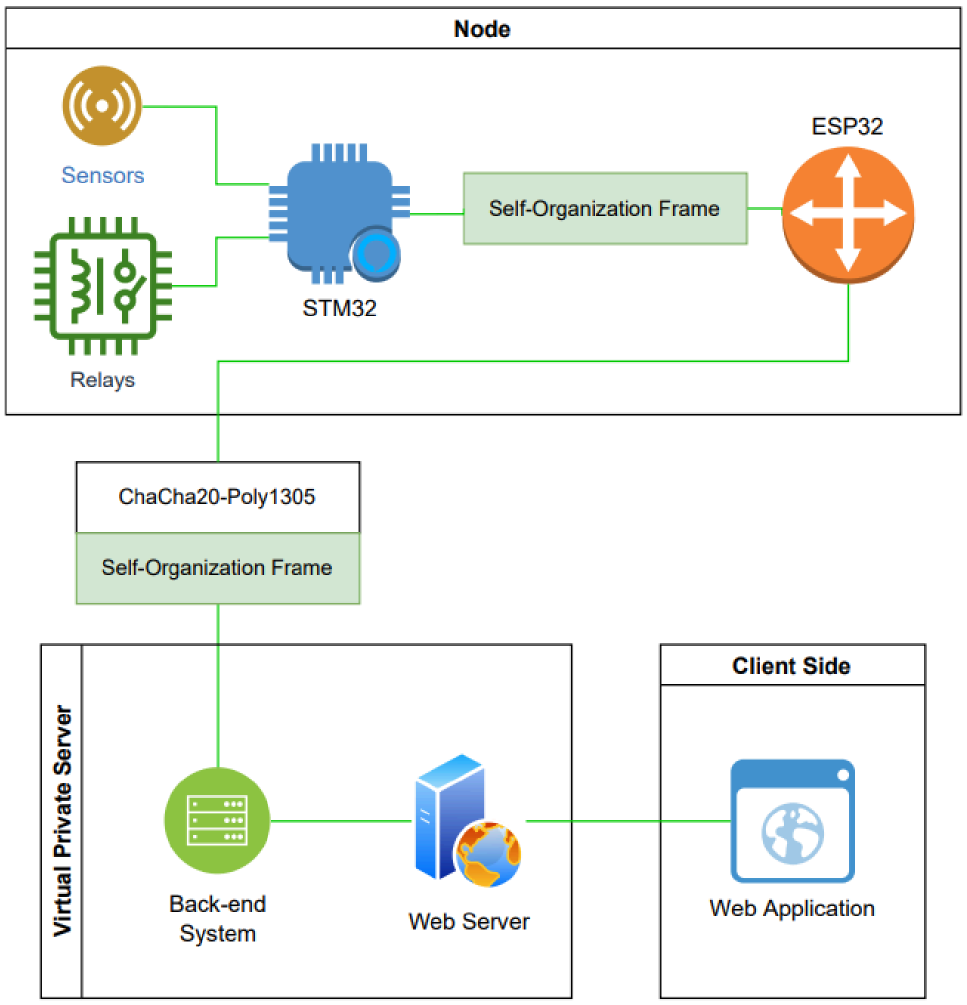
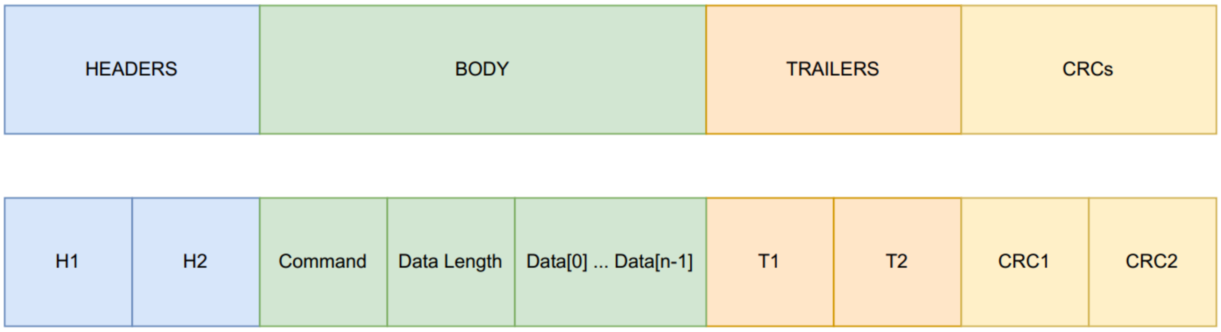


Fig. 5: Implementation of ChaCha20-Poly1305 and Data Framing on the IoT System

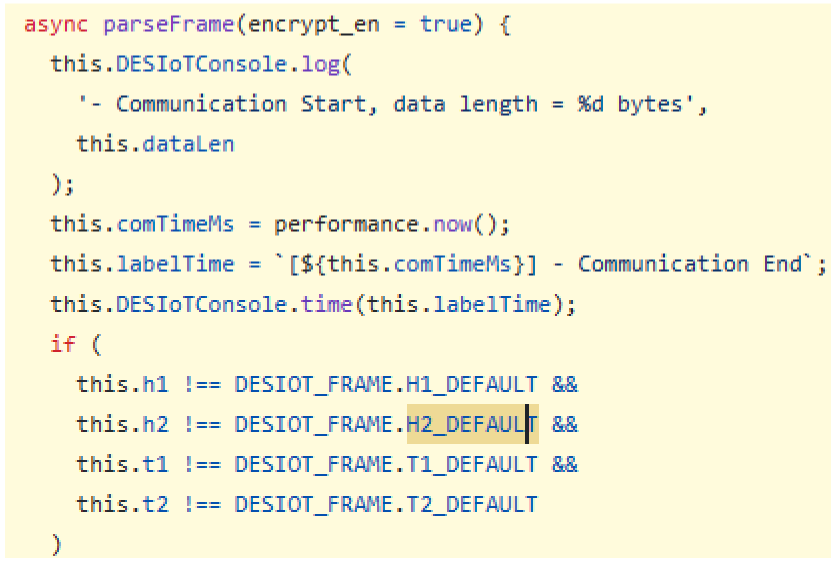
4. Frame Protocol: Structure and Parsing



Hình 2.2: Cấu trúc frame của hệ thống.

4.1. Frame Parsing in the server-side

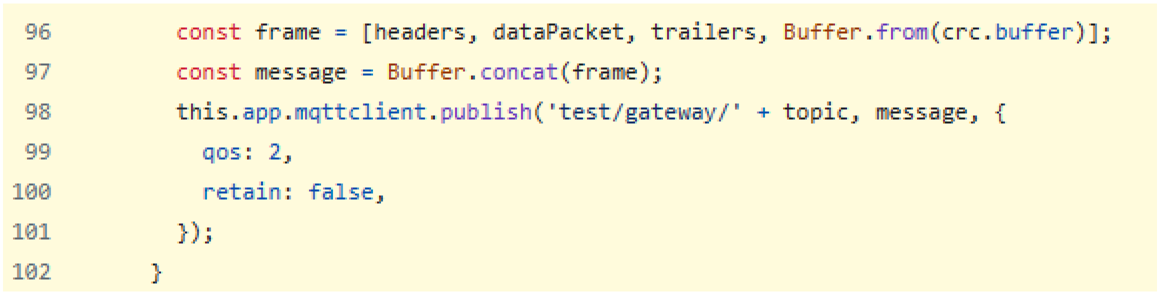
- [desiot-server/lib/src/frameHandler/index.js at 1.x.x · ngminhthanh12a3/desiot-server (github.com)]



```
async parseFrame(encrypt_en = true) {
  this.DESIoTConsole.log(
    '- Communication Start, data length = %d bytes',
    this.dataLen
  );
  this.comTimeMs = performance.now();
  this.labelTime = `[${this.comTimeMs}] - Communication End`;
  this.DESIoTConsole.time(this.labelTime);
  if (
    this.h1 !== DESIOT_FRAME.H1_DEFAULT &&
    this.h2 !== DESIOT_FRAME.H2_DEFAULT &&
    this.t1 !== DESIOT_FRAME.T1_DEFAULT &&
    this.t2 !== DESIOT_FRAME.T2_DEFAULT
  )
```

4.2. Frame Composing from the Server

- Before sending a frame to the ESP32 Gateway, the server constructs the frame components following the frame structure.



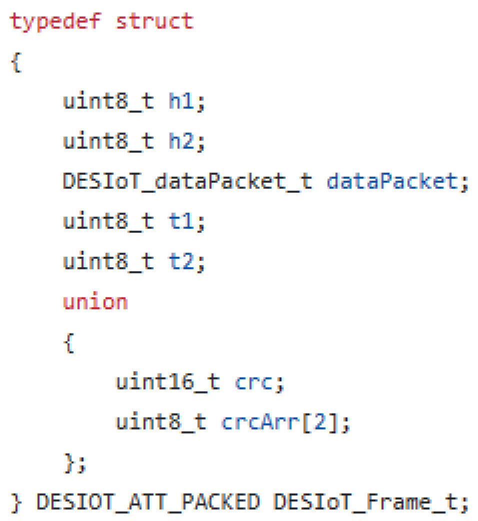
```
        const frame = [headers, dataPacket, trailers, Buffer.from(crc.buffer)];
        const message = Buffer.concat(frame);
        this.app.mqttclient.publish('test/gateway/' + topic, message, {
          qos: 2,
          retain: false,
        });
      }
```

4.3.

- ○ [desiot-server/lib/utils/DevSyncFrame.js at 1.x.x · ngminhthanh12a3/desiot-server (github.com)]

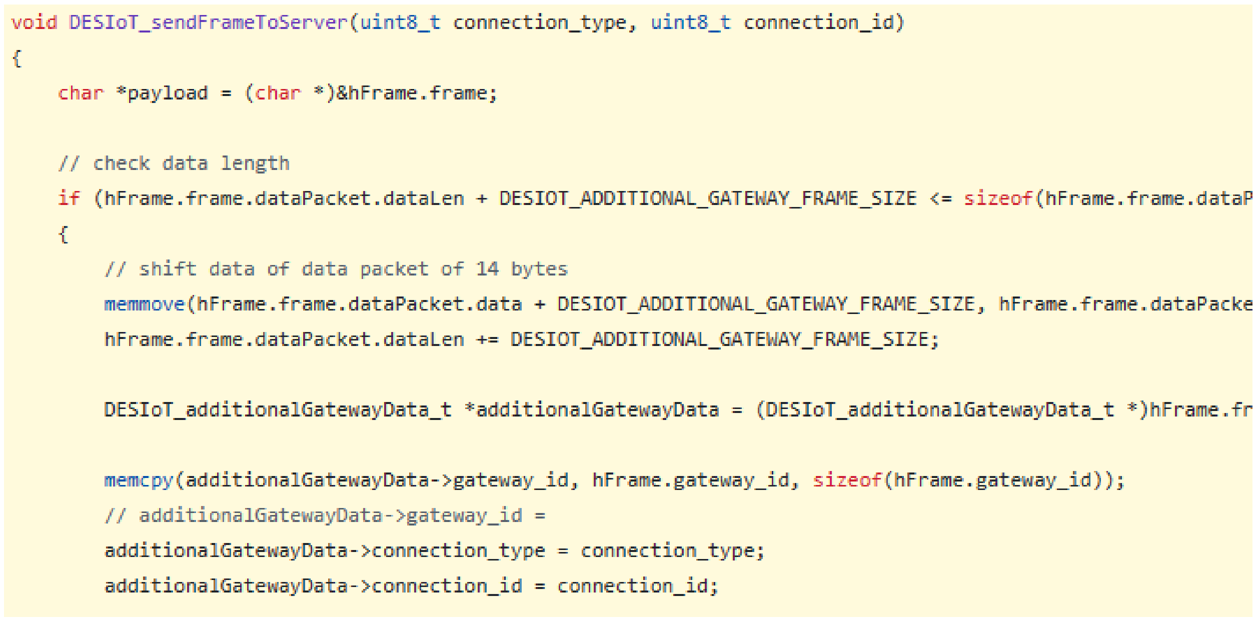
4.4. Frame Composing from the ESP32 Gateway

- The frame structure definition of the hardware:
  - ○ [DESIoT_ESP32_Gateway/include/DESIoT_Gateway.h at 1.x.x · ngminhthanh12a3/DESIoT_ESP32_Gateway (github.com)]



```
typedef struct
{
    uint8_t h1;
    uint8_t h2;
    DESIoT_dataPacket_t dataPacket;
    uint8_t t1;
    uint8_t t2;
    union
    {
        uint16_t crc;
        uint8_t crcArr[2];
    };
} DESIOT_ATT_PACKED DESIoT_Frame_t;
```

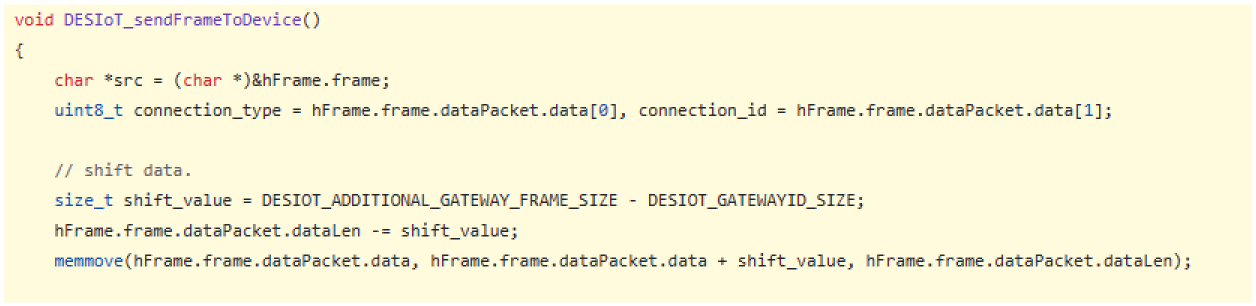
- The composing function manually constructs a frame before sending it to the server:



```
void DESIoT_sendFrameToServer(uint8_t connection_type, uint8_t connection_id)
{
    char *payload = (char *)&hFrame.frame;

    // check data length
    if (hFrame.frame.dataPacket.dataLen + DESIOT_ADDITIONAL_GATEWAY_FRAME_SIZE <= sizeof(hFrame.frame.dataP
    {
        // shift data of data packet of 14 bytes
        memmove(hFrame.frame.dataPacket.data + DESIOT_ADDITIONAL_GATEWAY_FRAME_SIZE, hFrame.frame.dataPacke
        hFrame.frame.dataPacket.dataLen += DESIOT_ADDITIONAL_GATEWAY_FRAME_SIZE;

        DESIoT_additionalGatewayData_t *additionalGatewayData = (DESIoT_additionalGatewayData_t *)hFrame.fr

        memcpy(additionalGatewayData->gateway_id, hFrame.gateway_id, sizeof(hFrame.gateway_id));
        // additionalGatewayData->gateway_id =
        additionalGatewayData->connection_type = connection_type;
        additionalGatewayData->connection_id = connection_id;
```

- ○ DESIoT_ESP32_Gateway/src/DESIoT_Gateway.cpp at 1.x.x · ngminhthanh12a3/DESIoT_ESP32_Gateway (github.com)

- The composing function manually constructs a frame before sending it to the hardware:



```
void DESIoT_sendFrameToDevice()
{
    char *src = (char *)&hFrame.frame;
    uint8_t connection_type = hFrame.frame.dataPacket.data[0], connection_id = hFrame.frame.dataPacket.data[1];

    // shift data.
    size_t shift_value = DESIOT_ADDITIONAL_GATEWAY_FRAME_SIZE - DESIOT_GATEWAYID_SIZE;
    hFrame.frame.dataPacket.dataLen -= shift_value;
    memmove(hFrame.frame.dataPacket.data, hFrame.frame.dataPacket.data + shift_value, hFrame.frame.dataPacket.dataLen);
```

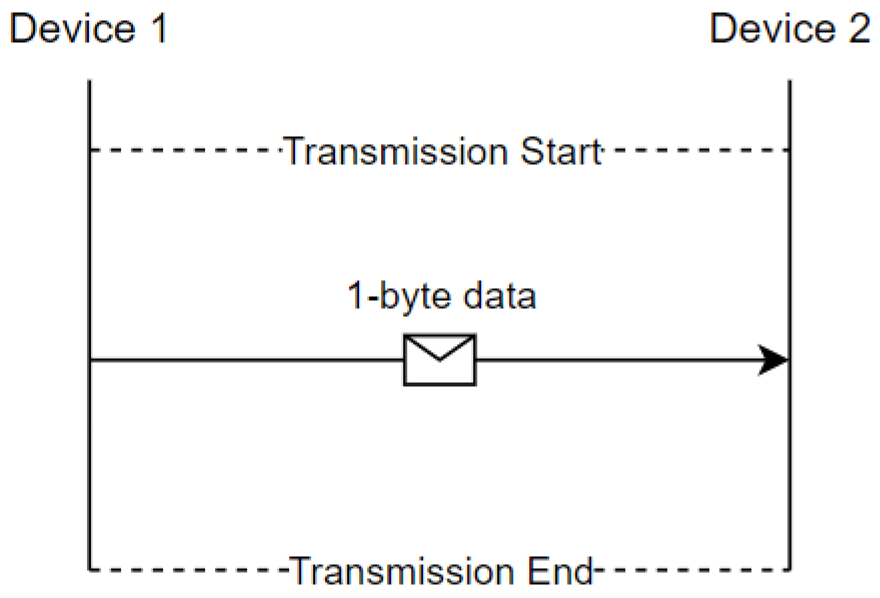
- DESIoT_ESP32_Gateway/src/DESIoT_Gateway.cpp at 1.x.x · ngminhthanh12a3/DESIoT_ESP32_Gateway (github.com)

5. Frame Protocol: Use Cases and Profound Issue

5.1. UART's Typical Cases

5.1.1. Simple Transmission: only-1-byte transmission

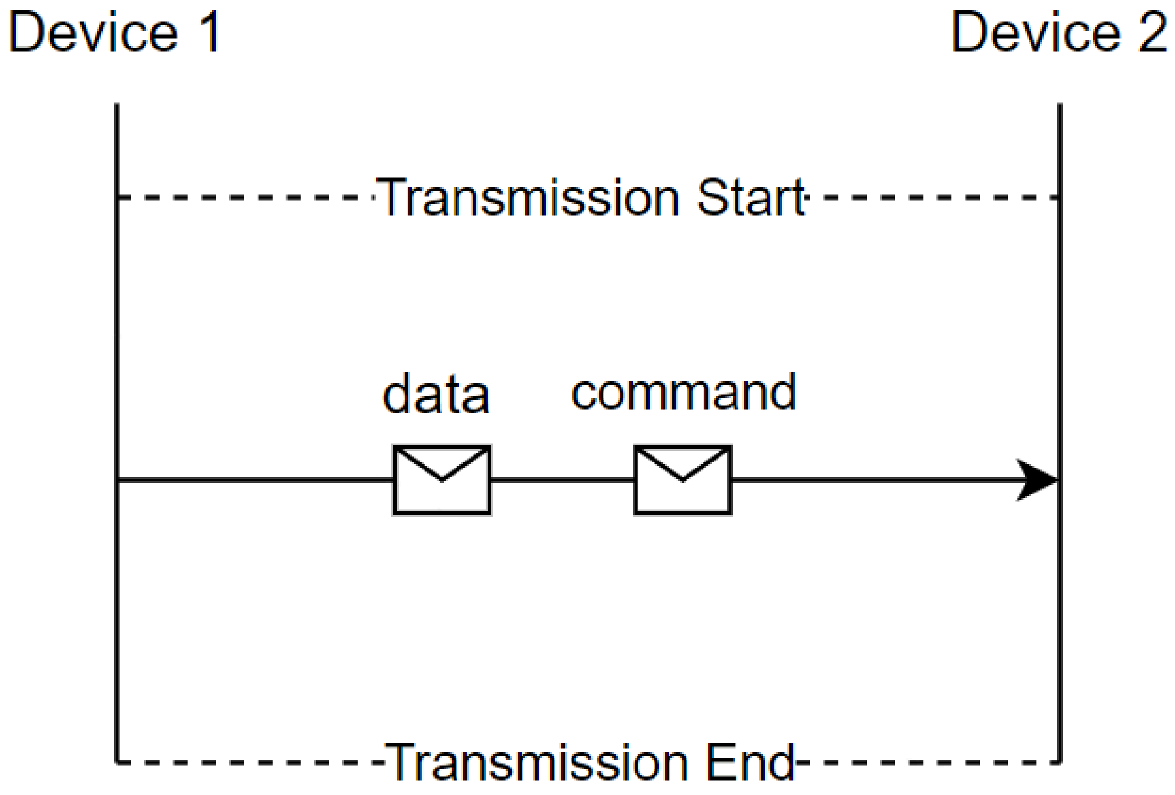
- ○ This is a typical case for beginners' approach.



- This case can be used to test the UART operation between 2 MCU.
- Communication operations between 2 MCU take place only based on 1-byte transmission data.
- For example, when a device receives the 1-byte transmission data, the device assigns the LED state to the 1-byte data value.

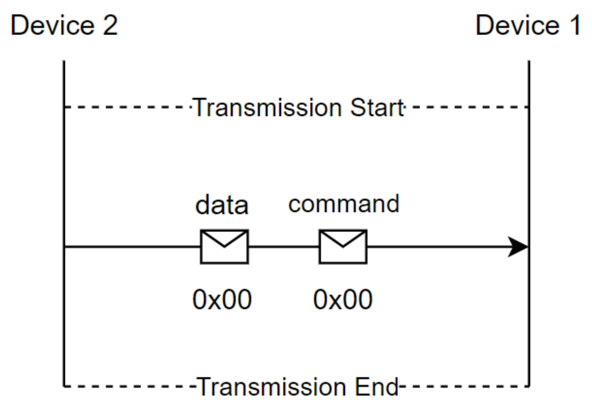
1.1.1. Simple Transmission: 2-byte transmission

- Another simple case of using UART is 2-byte communication.
- The operation is similar to the 1-byte communication.
- But the 1-byte transmission data is indexed by a 1-byte command.

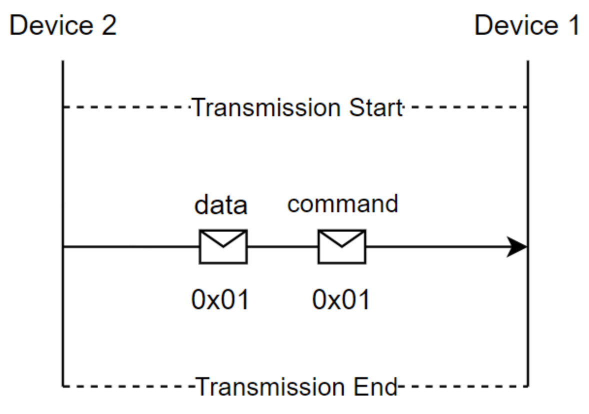


- The command is used to consider what exact operation is the transmission data used for.
- For example, an operation of transmission when the "Device 1" gets the LED or Switch state of the "Device 2"

Command	**Operation**
0x00	Device 1 gets the current LED state of Device 2
0x01	Device 1 gets the current Switch state of Device 2



The LED of Device 2 is off

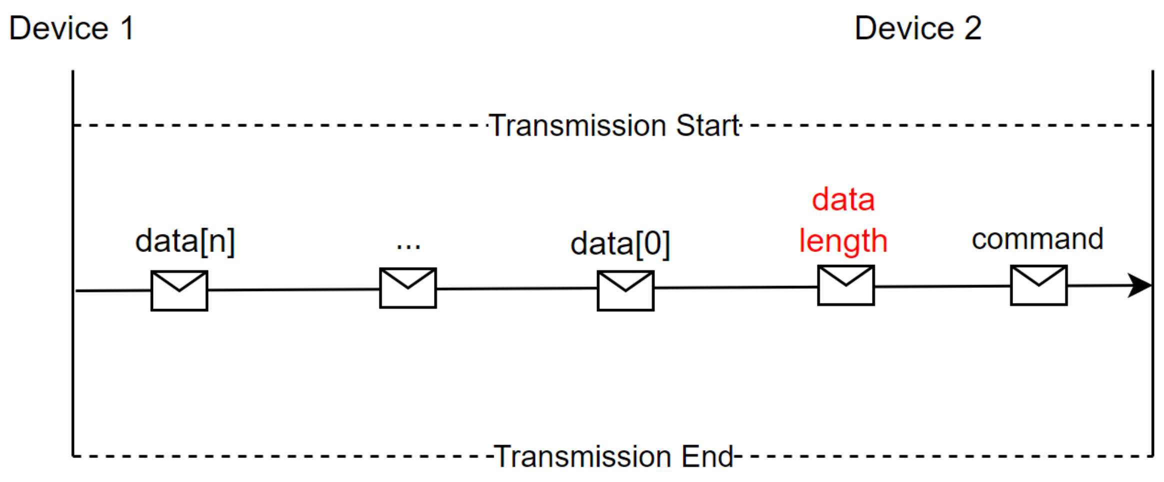


The Switch of Device 2 is on

- So, this is an example of using the "indexing" technique in a serial protocol.

5.1.2. Other Cases of Using Indexing Technique

- The usage of the indexing technique can be applied to complicated transactions requiring multiple-byte data for transmission.
- The number of data bytes is managed by the "data length" byte.

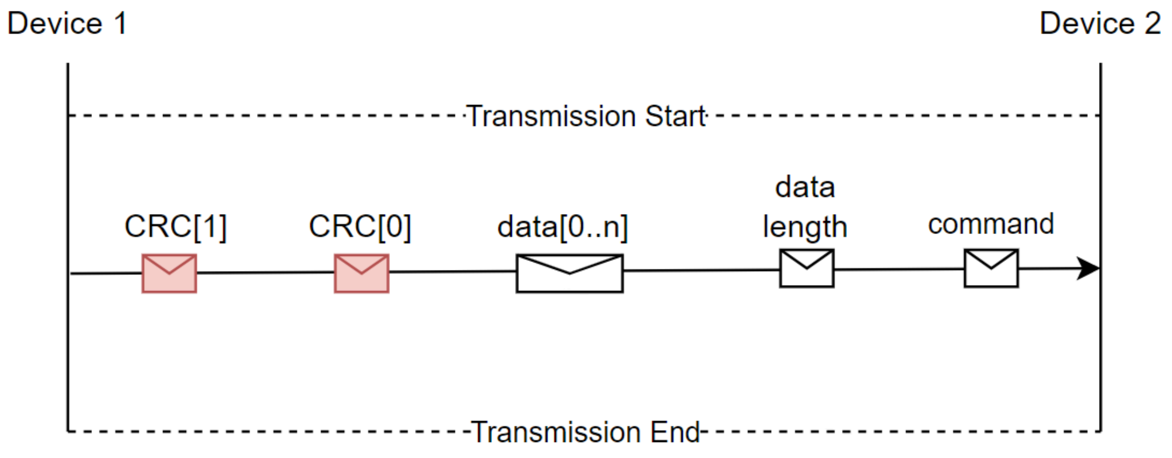


- So, this is a simple example of using the data management technique in transmission.

5.1.3. Case of Error Detection Technique

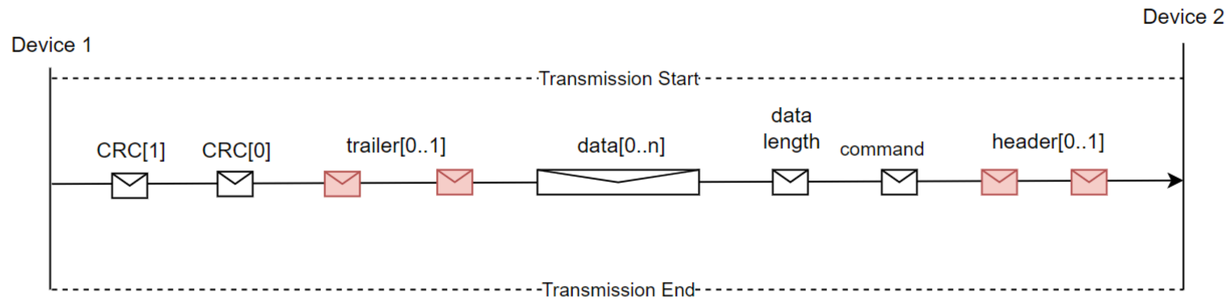
- The UART may perform correctly in short-term monitoring or operations.
- However, in long-term monitoring and operations, UART transmission is not always reliable and correct.

- So, how to deal with wrong-data value?
- The origin UART itself doesn't have any actual mechanism to detect errors.
- The Cyclic redundancy check (CRC) is a common method to detect errors in serial protocols such as UART.
- An example of using CRC is inserting 2 additional bytes to transmission to perform 16-bit CRC.



5.1.4. Case of Data Framing Technique

- In data transmission, there are several profound problems:
  1. Identification: how to determine whether a device is connecting with another reliable device in the same system or an external alias device?
  2. Configuration and synchronization: how to determine whether a device is contacting with another device having the same settings such as CRC type (8-, 16-, or 32-bit), command design,…?
  3. Security: how to determine that the transmission of the system is private and isolated on compared to other systems?
     Reference: UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices
- Data framing is the typical technique to provide identification, synchronization, and security to a transmission protocol.
- The normal usage of data framing is inserting additional headers and trailers to the transmission data.
- For example, we can insert a 2-byte header and trailer respectively to the start and end of a data package.



6. Hardware Implementation of the ESP32 Gateway

7. Lightweight Cryptography Implementation

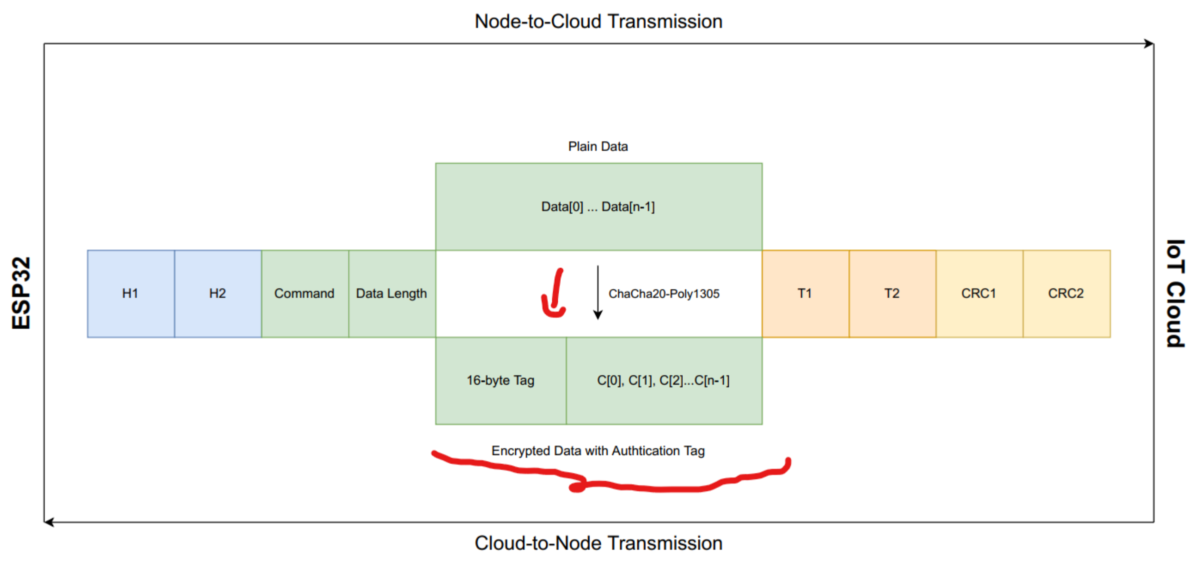

Node-to-Cloud Transmission
ESP32
Plain Data
Data[0] ... Data[n-1]
H1
H2
Command
Data Length
ChaCha20-Poly1305
T1
T2
CRC1
CRC2
IoT Cloud
16-byte Tag
C[0], C[1], C[2]...C[n-1]
Encrypted Data with Authtication Tag
Cloud-to-Node Transmission

Fig. 2: The ChaCha20-Poly1305 Implementation on the Proposed Frame Protocol

8. Setup Server

- Test the system in your local VM server.

8.1. Download the source code

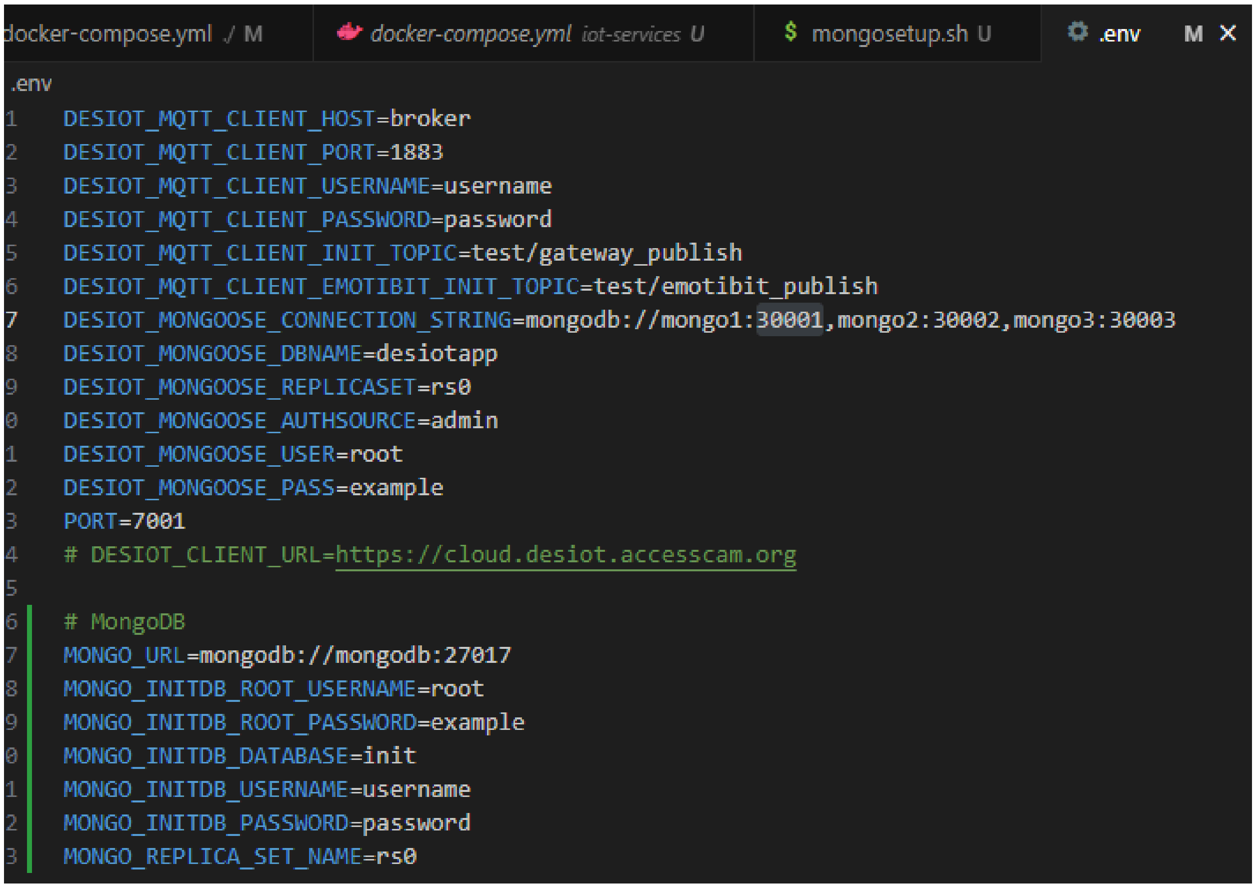
- desiot@desiot:~/desiot-server/testdir/desiot-server$ **git clone --branch QT-Demo https://github.com/ngminhthanh12a3/desiot-server.git**

8.2. Setup Database Private Key

- install make in ubuntu - Tìm trên Google

    - "sudo apt-get -y install make"
- Run the following command to initialize the database key:
    - desiot@desiot:~/desiot-server$ **make mongo-key-init**
- Start the system
    - desiot@desiot:~/desiot-server$ **make dev-up**

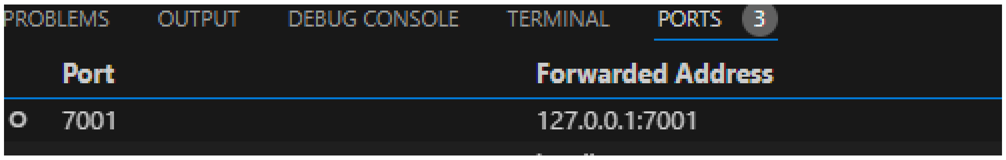
8.3. Server configuration environment



```
.env
DESIOT_MQTT_CLIENT_HOST=broker
DESIOT_MQTT_CLIENT_PORT=1883
DESIOT_MQTT_CLIENT_USERNAME=username
DESIOT_MQTT_CLIENT_PASSWORD=password
DESIOT_MQTT_CLIENT_INIT_TOPIC=test/gateway_publish
DESIOT_MQTT_CLIENT_EMOTIBIT_INIT_TOPIC=test/emotibit_publish
DESIOT_MONGOOSE_CONNECTION_STRING=mongodb://mongo1:30001,mongo2:30002,mongo3:30003
DESIOT_MONGOOSE_DBNAME=desiotapp
DESIOT_MONGOOSE_REPLICASET=rs0
DESIOT_MONGOOSE_AUTHSOURCE=admin
DESIOT_MONGOOSE_USER=root
DESIOT_MONGOOSE_PASS=example
PORT=7001
# DESIOT_CLIENT_URL=https://cloud.desiot.accesscam.org

# MongoDB
MONGO_URL=mongodb://mongodb:27017
MONGO_INITDB_ROOT_USERNAME=root
MONGO_INITDB_ROOT_PASSWORD=example
MONGO_INITDB_DATABASE=init
MONGO_INITDB_USERNAME=username
MONGO_INITDB_PASSWORD=password
MONGO_REPLICA_SET_NAME=rs0
```

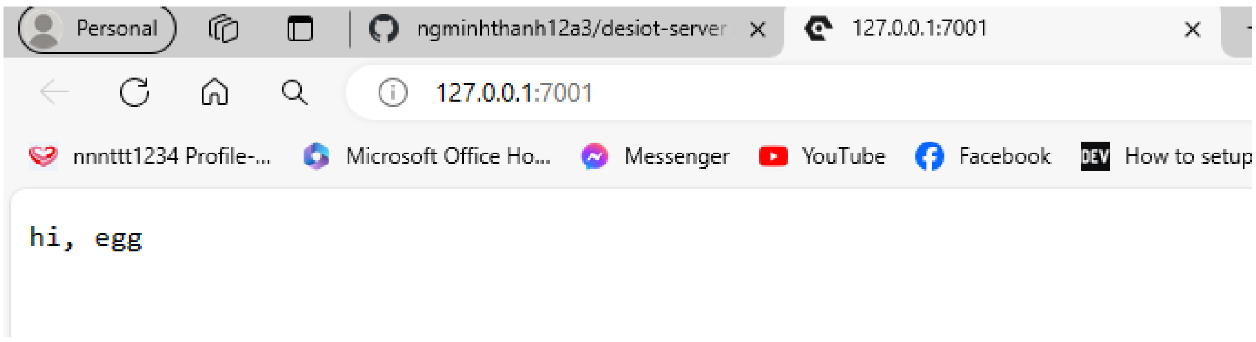
- Change the configuration environment in the ".env" file if you want to change the **port** of the broker or server.

8.4. Test the Server

- 

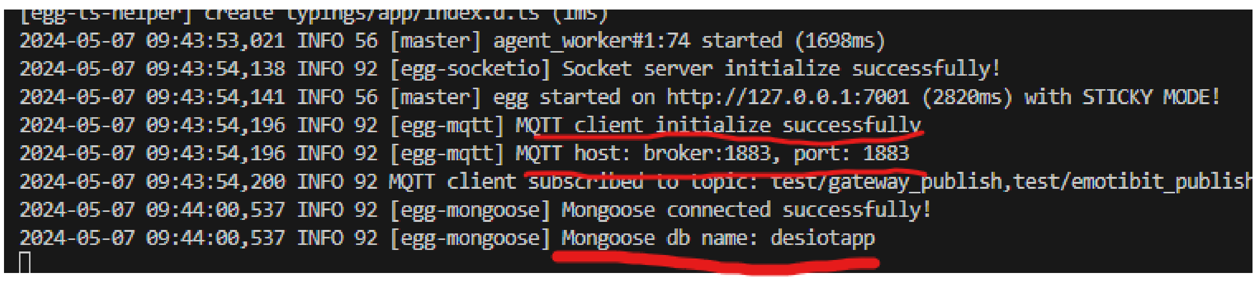
```
Port     Forwarded Address
7001     127.0.0.1:7001
```

    - Setup the port forwarding.



```
hi, egg
```

-
- View server logs for checking the successful configurations of MQTT Broker and MongoDB connections
  - desiot@desiot:~/desiot-server$ **docker logs -f desiot-server-desiot-server-1**



```
[egg-ts-helper] create typings/app/index.d.ts (1ms)
2024-05-07 09:43:53,021 INFO 56 [master] agent_worker#1:74 started (1698ms)
2024-05-07 09:43:54,138 INFO 92 [egg-socketio] Socket server initialize successfully!
2024-05-07 09:43:54,141 INFO 56 [master] egg started on http://127.0.0.1:7001 (2820ms) with STICKY MODE!
2024-05-07 09:43:54,196 INFO 92 [egg-mqtt] MQTT client initialize successfully
2024-05-07 09:43:54,196 INFO 92 [egg-mqtt] MQTT host: broker:1883, port: 1883
2024-05-07 09:43:54,200 INFO 92 MQTT client subscribed to topic: test/gateway_publish,test/emotibit_publish
2024-05-07 09:44:00,537 INFO 92 [egg-mongoose] Mongoose connected successfully!
2024-05-07 09:44:00,537 INFO 92 [egg-mongoose] Mongoose db name: desiotapp
```

-
- Re-run the system if any error occur:
  - desiot@desiot:~/desiot-server$ **make dev-reup**
-