HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

IoT, Microcontrollers
Autumn 2022
Teacher: Timo Karppinen
Student: Eduardo Luis Hofmann

1 Introduction

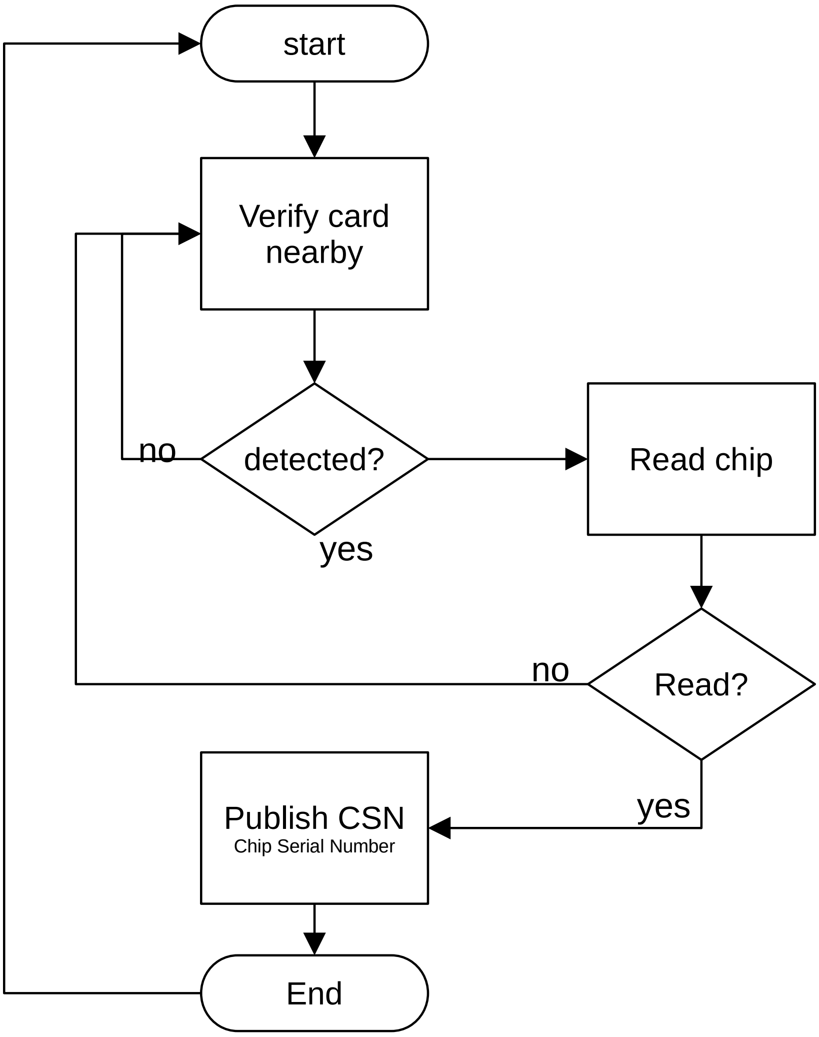
As products become more expensive or require special control, including traceability. A robust solution is needed to collect a large amount of data generated from several collection points, from begin to end in a production process. Nevertheless, such solution must be easy to narrow down to a single unit. Some examples of manufacturing process that benefit from a product tracking system:
- Meat processor, tracking from cattle ear tag to single meat package using RFID tag.
- Security documents factory, from the moment components become an unfinished product, a RFID tag or in case the product has a contactless chip it can be used to track units during the manufacturing process.
- Assembly line, collecting data from components already mounted

1.1 Product idea

RFID reader publishing CSN (chip serial number) to MQTT broker.

1.2 Use case

Security documents factory, Collecting CSN between two Process. That is used daily to verify that all products were consumed. This solution provide actionable data for people overseeing production.



2 Concept

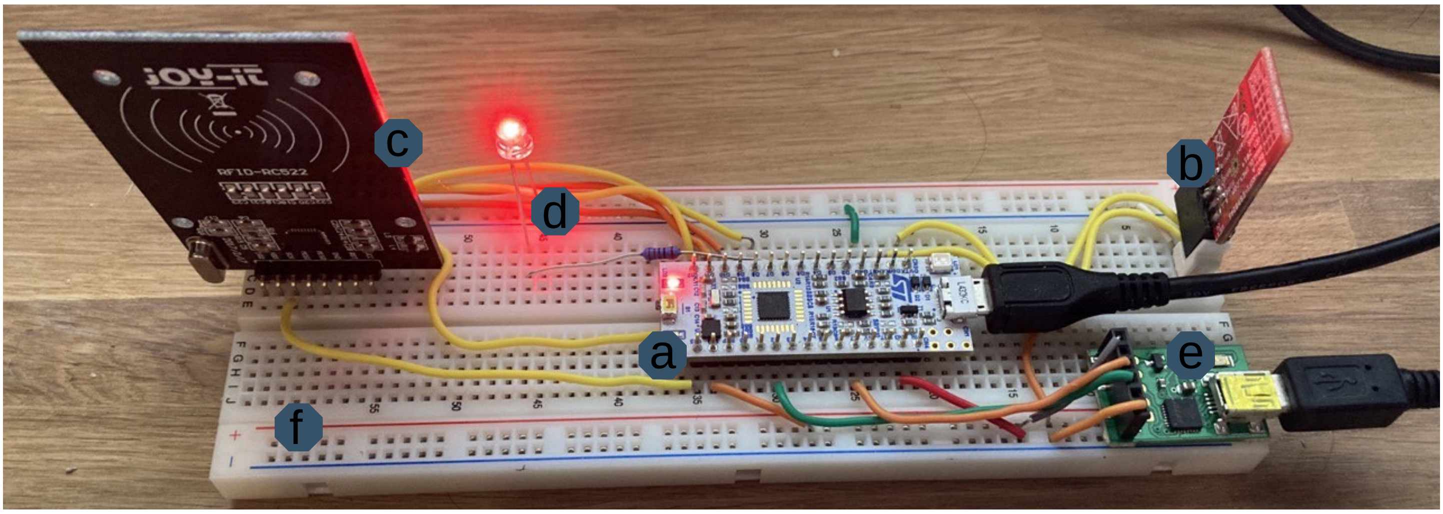
The completed solution use components available in the consumer market for development and testing of the technology of Embedded computer, WiFI and RFID.

2.1 Process description

The code executed in a micro-controller Nucleo L432KC continuously check if a chip is in the reading zone of a RFID RC522 reader. When a chip is identified, a read command is triggered to collect the CSN, chip serial number. The CSN is published to a MQTT broker, using a WiFi connection.

2.2 Components

a	microcontroller Nucleo L432KC
b	WiFi Module MOD-WIFI-ESP8266
c	RFID RC522
d	Red LED
e	DPI - Debug Port Interface/Serial to USB
f	Breadboard and cables



2.3 Specifications

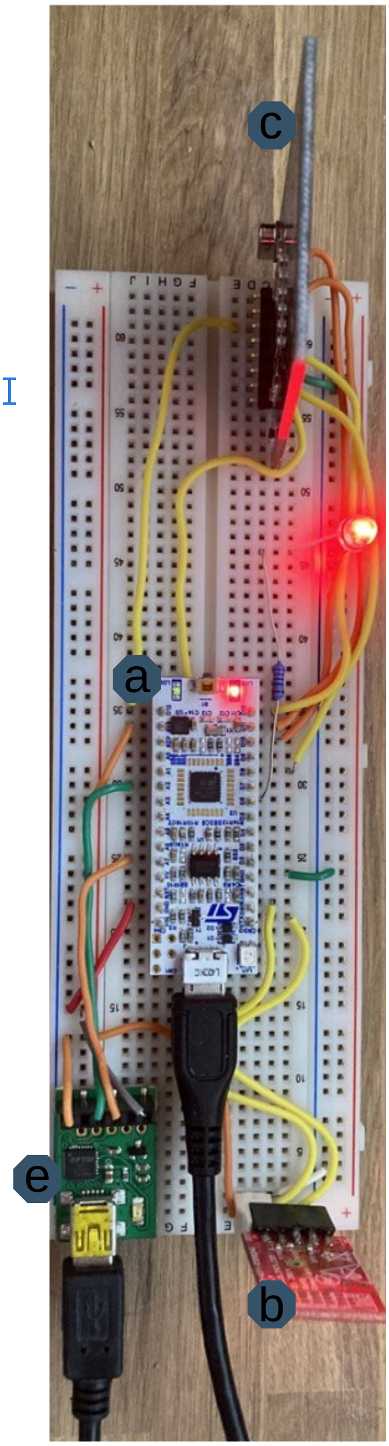
The detailed specification for the main components of the solution can be found on vendor web pages.

a microcontroller Nucleo L432KC
https://os.mbed.com/platforms/ST-Nucleo-L432KC/

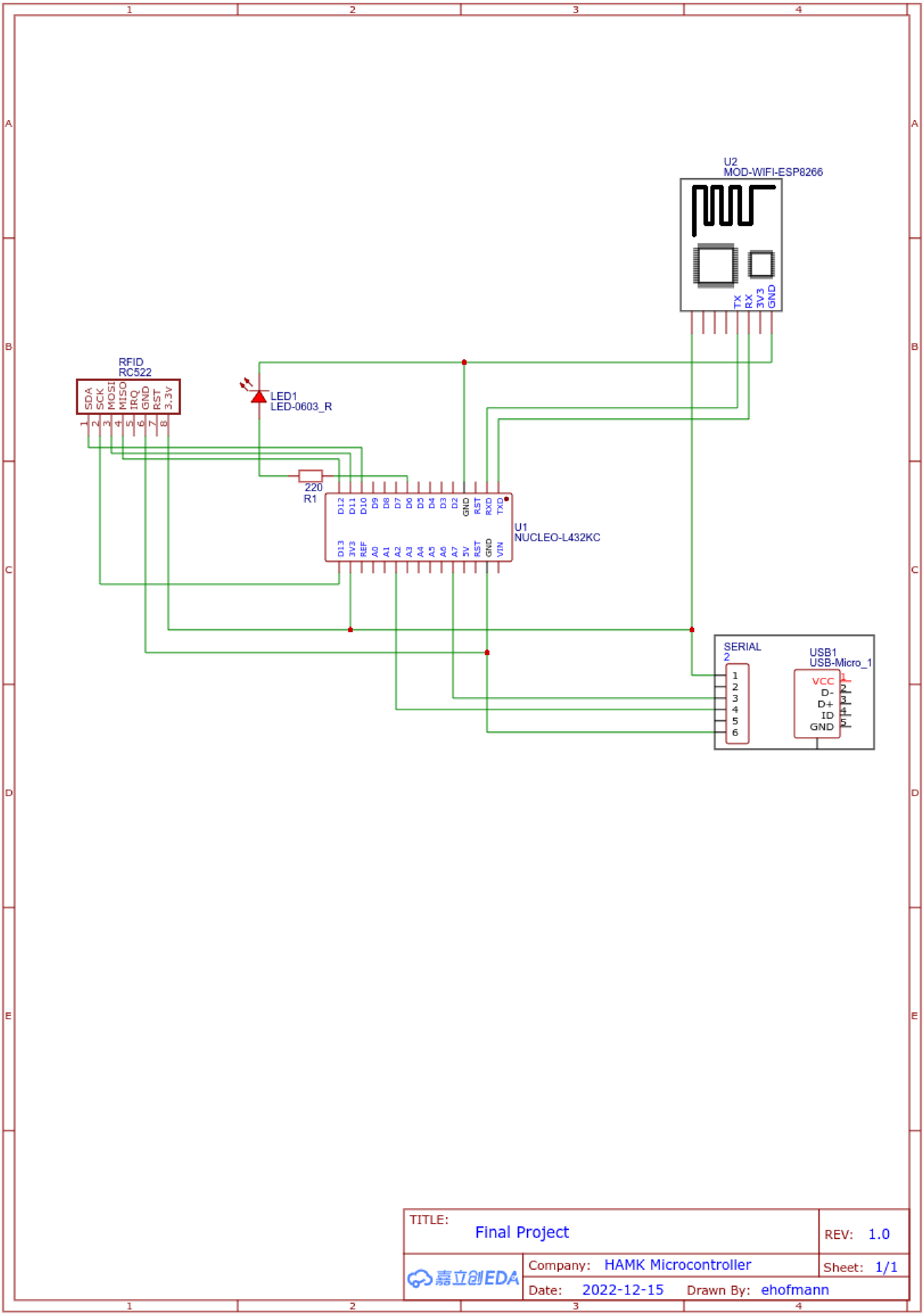
b WiFi Module MOD-WIFI-ESP8266
https://www.olimex.com/Products/IoT/ESP8266/MOD-WIFI-ESP8266/open-source-hardware

c RFID RC522
https://joy-it.net/en/products/SBC-RFID-RC522

e DPI - Debug Port Interface/Serial to USB
https://www.acmesystems.it/DPI



2.4 Diagram



TITLE: Final Project

REV: 1.0

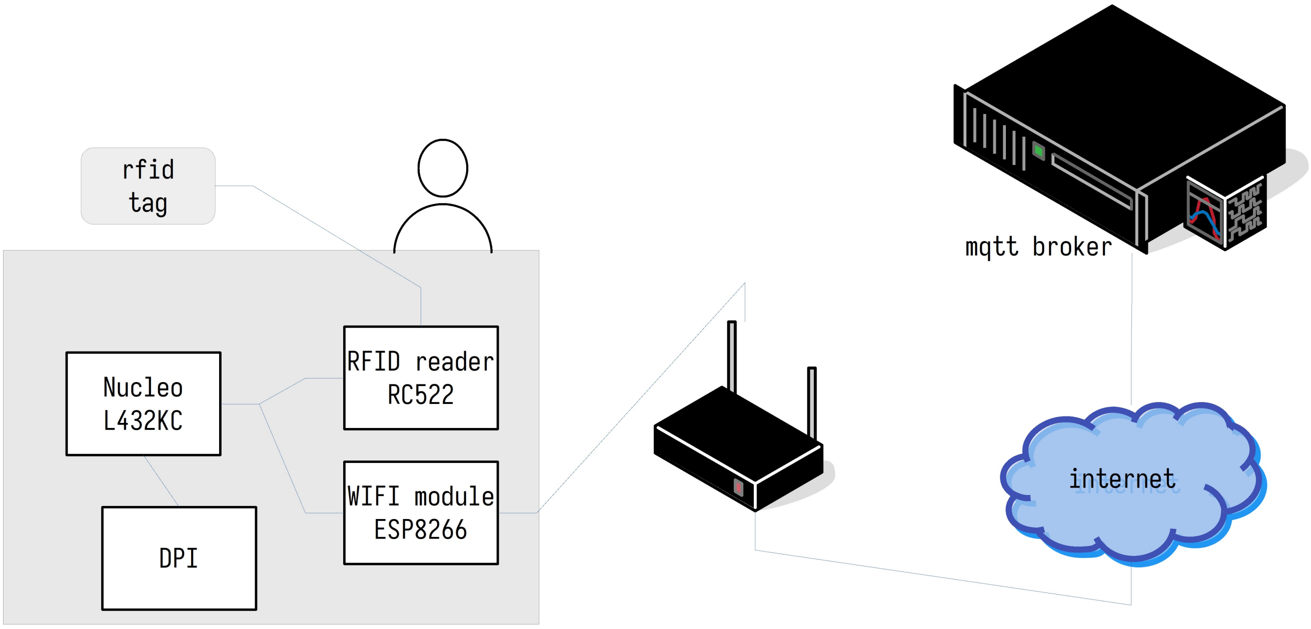
Company: HAMK Microcontroller

Sheet: 1/1

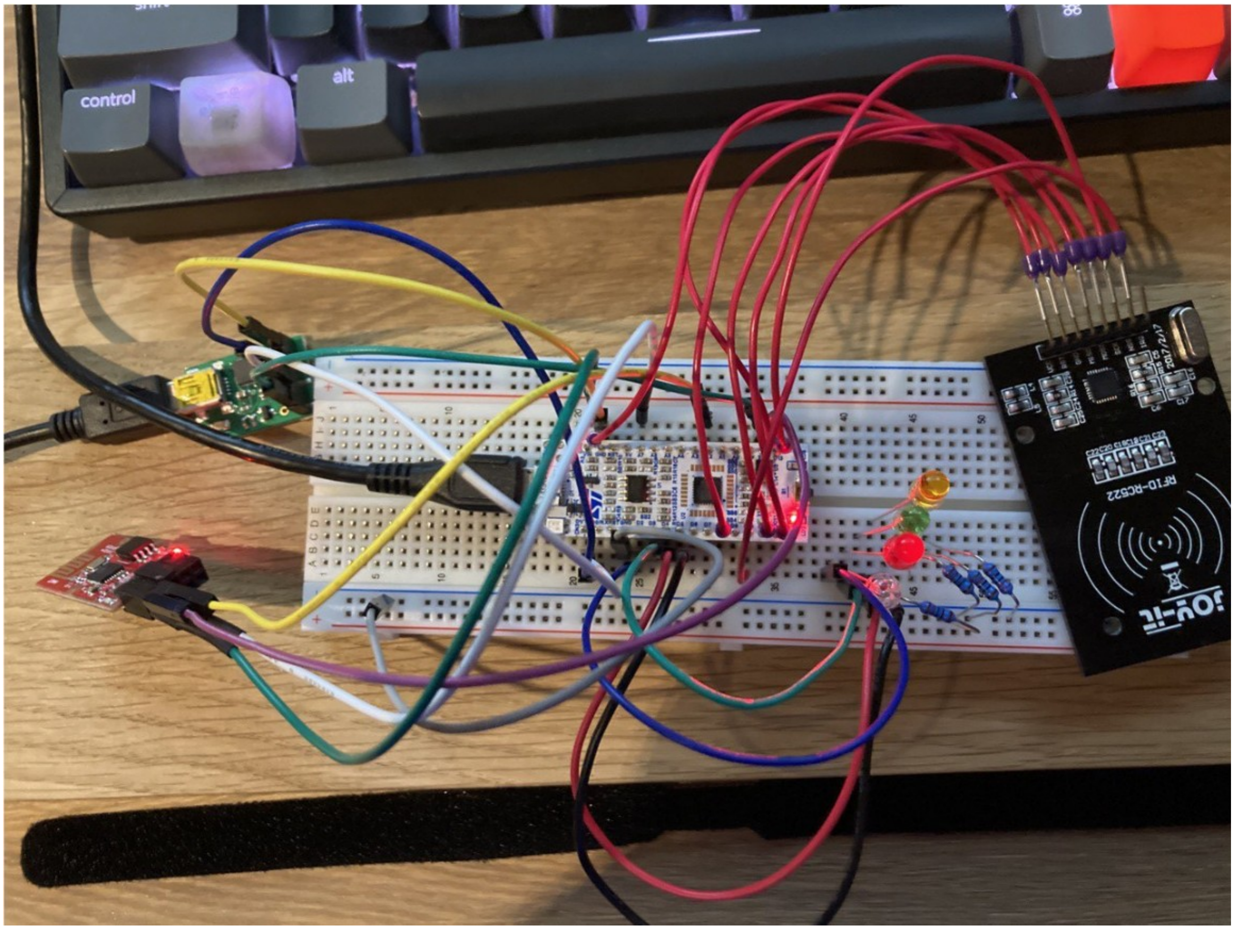
Date: 2022-12-15

Drawn By: ehofmann

3 Solution



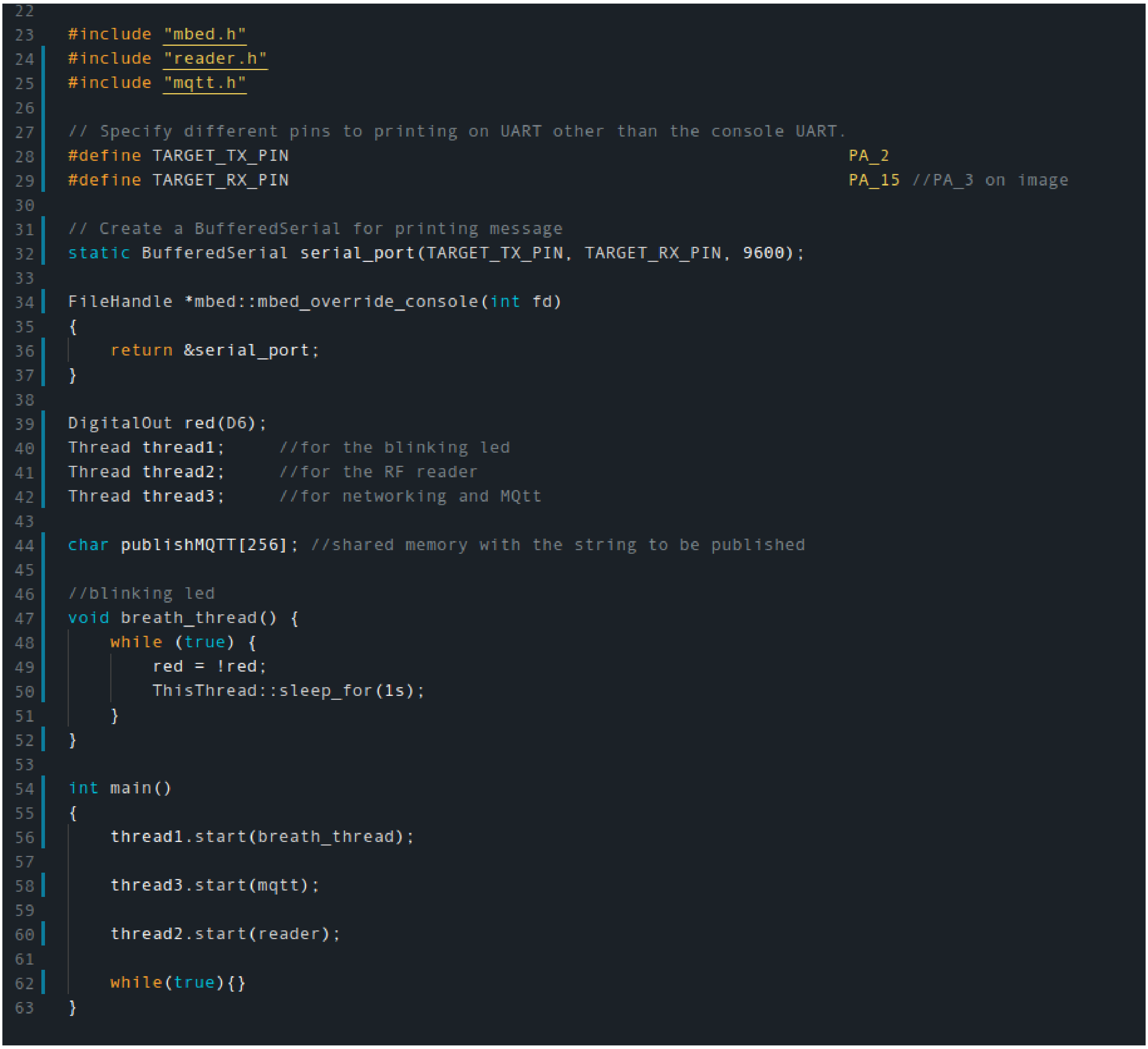
The RFID reader RC522 is connected to the microcontroller Nucleo L432KC through SPI (serial Peripheral Interface) and the WiFi module MOD-WIFI-ESP8266 is connected through serial interface. A DPI (debug Port Interface) that convert serial to USB is used for debug operation.



Code can be found here:
https://github.com/githubedu/HAMK-Microcontroller

3.1 main.cpp

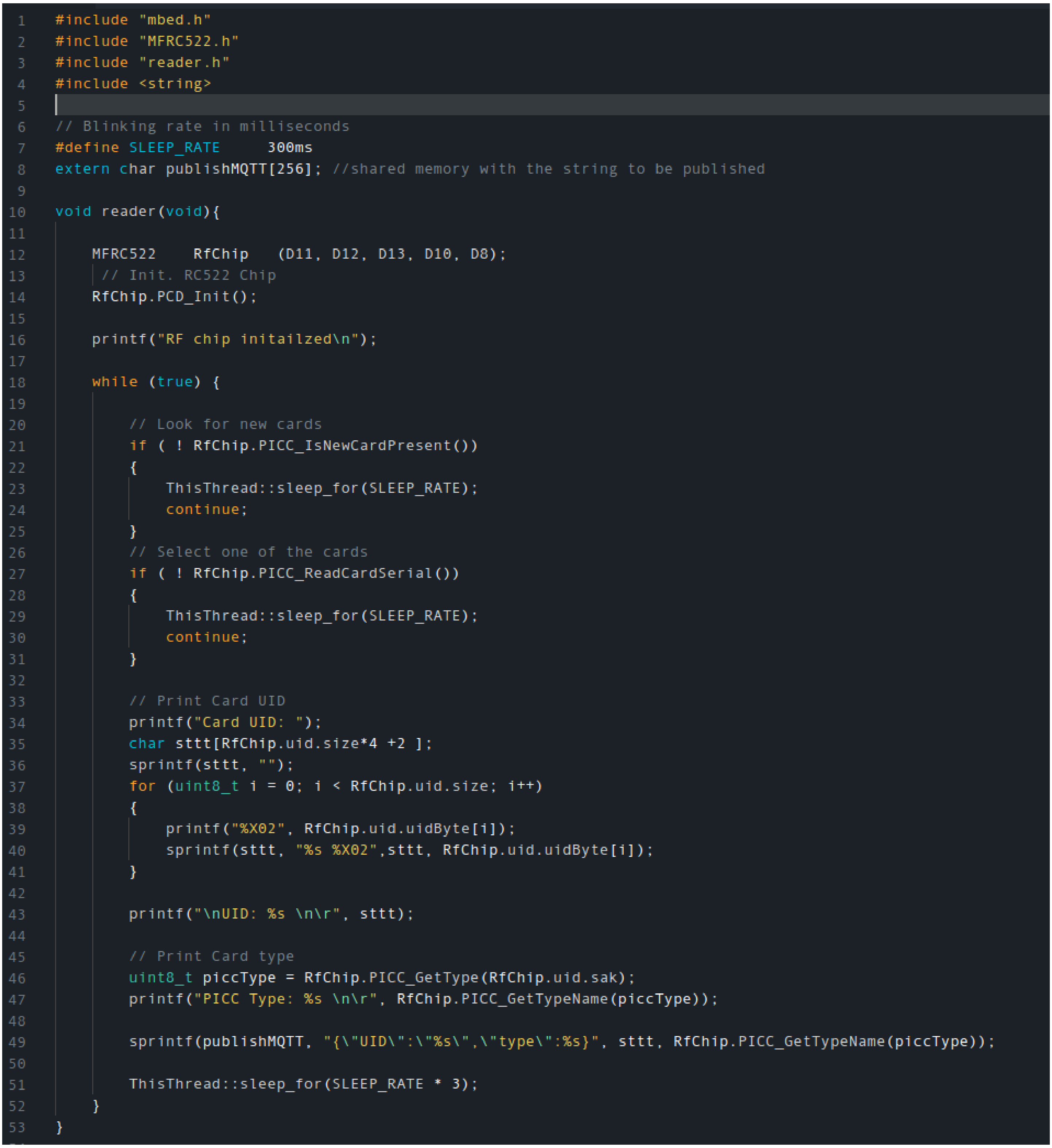
Beside MQTT, MFR522 and ESP8266 libraries forked from the community. The solution has three code files, main.cpp initialize the code and launch three threads. Being thread1 to control a blinking red led. thread2 for the RF reader and thread3 for networking and MQTT.



```cpp

#include "mbed.h"
#include "reader.h"
#include "mqtt.h"

// Specify different pins to printing on UART other than the console UART.
#define TARGET_TX_PIN                                                     PA_2
#define TARGET_RX_PIN                                                     PA_15 //PA_3 on image

// Create a BufferedSerial for printing message
static BufferedSerial serial_port(TARGET_TX_PIN, TARGET_RX_PIN, 9600);

FileHandle *mbed::mbed_override_console(int fd)
{
    return &serial_port;
}

DigitalOut red(D6);
Thread thread1;     //for the blinking led
Thread thread2;     //for the RF reader
Thread thread3;     //for networking and MQtt

char publishMQTT[256]; //shared memory with the string to be published

//blinking led
void breath_thread() {
    while (true) {
        red = !red;
        ThisThread::sleep_for(1s);
    }
}

int main()
{
    thread1.start(breath_thread);

    thread3.start(mqtt);

    thread2.start(reader);

    while(true){}
}
```

3.2 reader.cpp

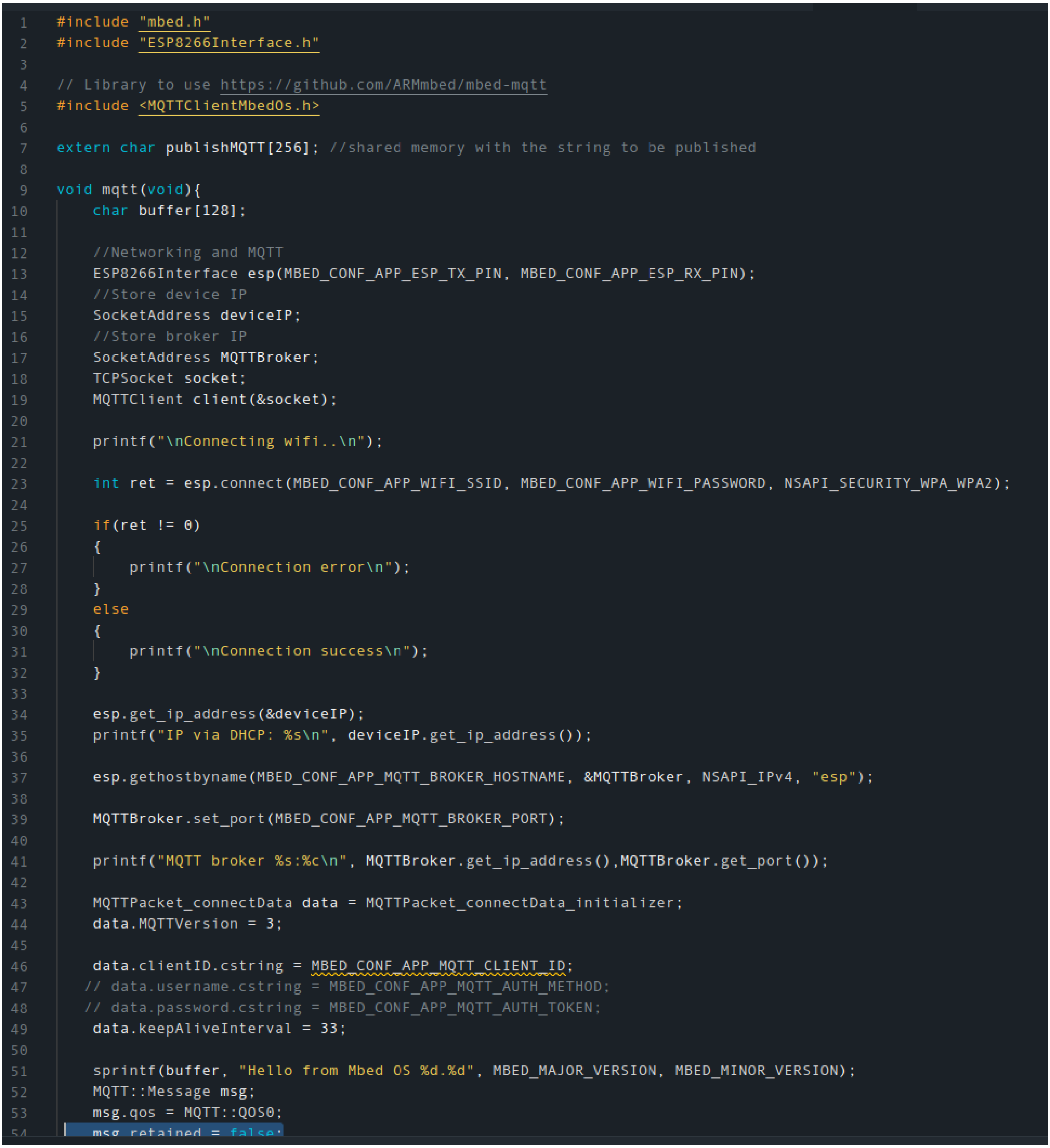
The code check continuously if a card is near the reader, when is present. It continue to read the CSN and the information is formatted to a global variable.



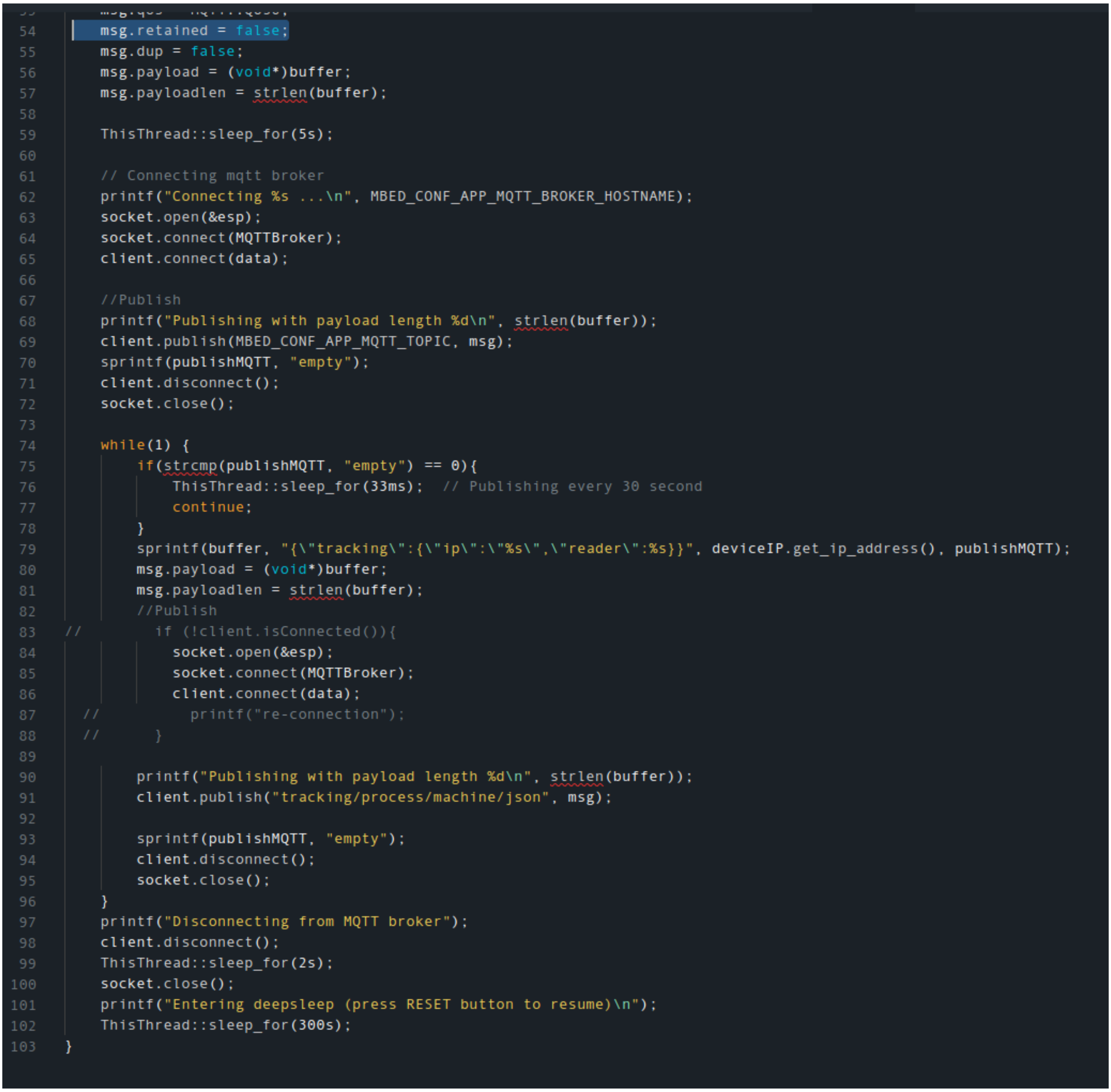
```cpp
#include "mbed.h"
#include "MFRC522.h"
#include "reader.h"
#include <string>

// Blinking rate in milliseconds
#define SLEEP_RATE     300ms
extern char publishMQTT[256]; //shared memory with the string to be published

void reader(void){

    MFRC522    RfChip   (D11, D12, D13, D10, D8);
    // Init. RC522 Chip
    RfChip.PCD_Init();

    printf("RF chip initailzed\n");

    while (true) {

        // Look for new cards
        if ( ! RfChip.PICC_IsNewCardPresent())
        {
            ThisThread::sleep_for(SLEEP_RATE);
            continue;
        }
        // Select one of the cards
        if ( ! RfChip.PICC_ReadCardSerial())
        {
            ThisThread::sleep_for(SLEEP_RATE);
            continue;
        }

        // Print Card UID
        printf("Card UID: ");
        char sttt[RfChip.uid.size*4 +2 ];
        sprintf(sttt, "");
        for (uint8_t i = 0; i < RfChip.uid.size; i++)
        {
            printf("%X02", RfChip.uid.uidByte[i]);
            sprintf(sttt, "%s %X02",sttt, RfChip.uid.uidByte[i]);
        }

        printf("\nUID: %s \n\r", sttt);

        // Print Card type
        uint8_t piccType = RfChip.PICC_GetType(RfChip.uid.sak);
        printf("PICC Type: %s \n\r", RfChip.PICC_GetTypeName(piccType));

        sprintf(publishMQTT, "{\"UID\":\"%s\",\"type\":%s}", sttt, RfChip.PICC_GetTypeName(piccType));

        ThisThread::sleep_for(SLEEP_RATE * 3);
    }
}
```

3.3 mqtt.cpp

The code to connect to WiFi network and publish the MQTT topic is in the file mqtt.cpp. It stay in a continuous loop, checking if there is a change in the global variable. When it happen, it will publish the data from the global variable and add the device IP address as identifier.



```
#include "mbed.h"
#include "ESP8266Interface.h"

// Library to use https://github.com/ARMmbed/mbed-mqtt
#include <MQTTClientMbedOs.h>

extern char publishMQTT[256]; //shared memory with the string to be published

void mqtt(void){
    char buffer[128];

    //Networking and MQTT
    ESP8266Interface esp(MBED_CONF_APP_ESP_TX_PIN, MBED_CONF_APP_ESP_RX_PIN);
    //Store device IP
    SocketAddress deviceIP;
    //Store broker IP
    SocketAddress MQTTBroker;
    TCPSocket socket;
    MQTTClient client(&socket);

    printf("\nConnecting wifi..\n");

    int ret = esp.connect(MBED_CONF_APP_WIFI_SSID, MBED_CONF_APP_WIFI_PASSWORD, NSAPI_SECURITY_WPA_WPA2);

    if(ret != 0)
    {
        printf("\nConnection error\n");
    }
    else
    {
        printf("\nConnection success\n");
    }

    esp.get_ip_address(&deviceIP);
    printf("IP via DHCP: %s\n", deviceIP.get_ip_address());

    esp.gethostbyname(MBED_CONF_APP_MQTT_BROKER_HOSTNAME, &MQTTBroker, NSAPI_IPv4, "esp");

    MQTTBroker.set_port(MBED_CONF_APP_MQTT_BROKER_PORT);

    printf("MQTT broker %s:%c\n", MQTTBroker.get_ip_address(),MQTTBroker.get_port());

    MQTTPacket_connectData data = MQTTPacket_connectData_initializer;
    data.MQTTVersion = 3;

    data.clientID.cstring = MBED_CONF_APP_MQTT_CLIENT_ID;
   // data.username.cstring = MBED_CONF_APP_MQTT_AUTH_METHOD;
   // data.password.cstring = MBED_CONF_APP_MQTT_AUTH_TOKEN;
    data.keepAliveInterval = 33;

    sprintf(buffer, "Hello from Mbed OS %d.%d", MBED_MAJOR_VERSION, MBED_MINOR_VERSION);
    MQTT::Message msg;
    msg.qos = MQTT::QOS0;
    msg.retained = false;
```

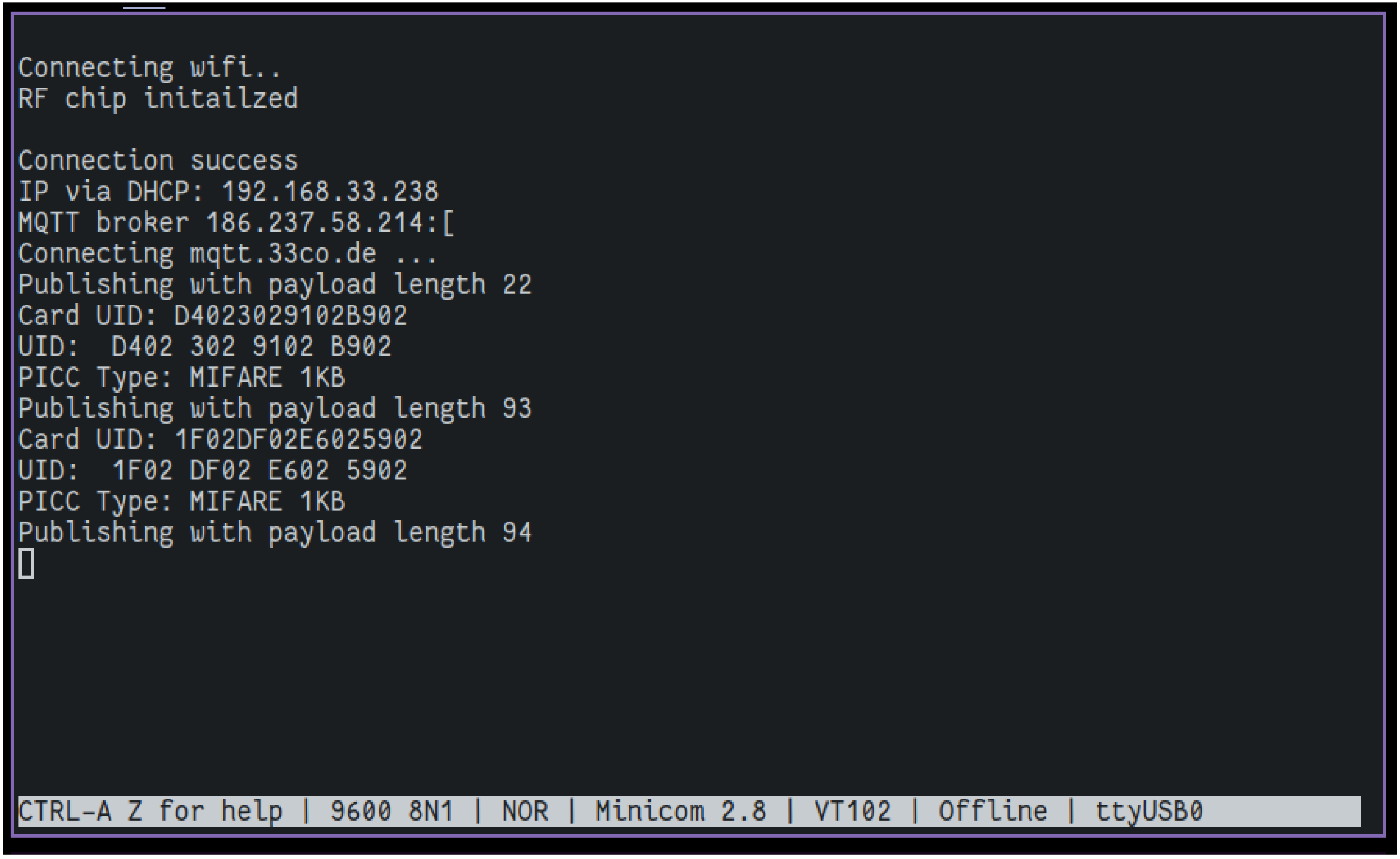


```
    msg.qos = MQTT::QOS0;
    msg.retained = false;
    msg.dup = false;
    msg.payload = (void*)buffer;
    msg.payloadlen = strlen(buffer);

    ThisThread::sleep_for(5s);

    // Connecting mqtt broker
    printf("Connecting %s ...\n", MBED_CONF_APP_MQTT_BROKER_HOSTNAME);
    socket.open(&esp);
    socket.connect(MQTTBroker);
    client.connect(data);

    //Publish
    printf("Publishing with payload length %d\n", strlen(buffer));
    client.publish(MBED_CONF_APP_MQTT_TOPIC, msg);
    sprintf(publishMQTT, "empty");
    client.disconnect();
    socket.close();

    while(1) {
        if(strcmp(publishMQTT, "empty") == 0){
            ThisThread::sleep_for(33ms);  // Publishing every 30 second
            continue;
        }
        sprintf(buffer, "{\"tracking\":{\"ip\":\"%s\",\"reader\":%s}}", deviceIP.get_ip_address(), publishMQTT);
        msg.payload = (void*)buffer;
        msg.payloadlen = strlen(buffer);
        //Publish
//        if (!client.isConnected()){
            socket.open(&esp);
            socket.connect(MQTTBroker);
            client.connect(data);
  //          printf("re-connection");
  //      }

        printf("Publishing with payload length %d\n", strlen(buffer));
        client.publish("tracking/process/machine/json", msg);

        sprintf(publishMQTT, "empty");
        client.disconnect();
        socket.close();
    }
    printf("Disconnecting from MQTT broker");
    client.disconnect();
    ThisThread::sleep_for(2s);
    socket.close();
    printf("Entering deepsleep (press RESET button to resume)\n");
    ThisThread::sleep_for(300s);
}
```

4 Analysis

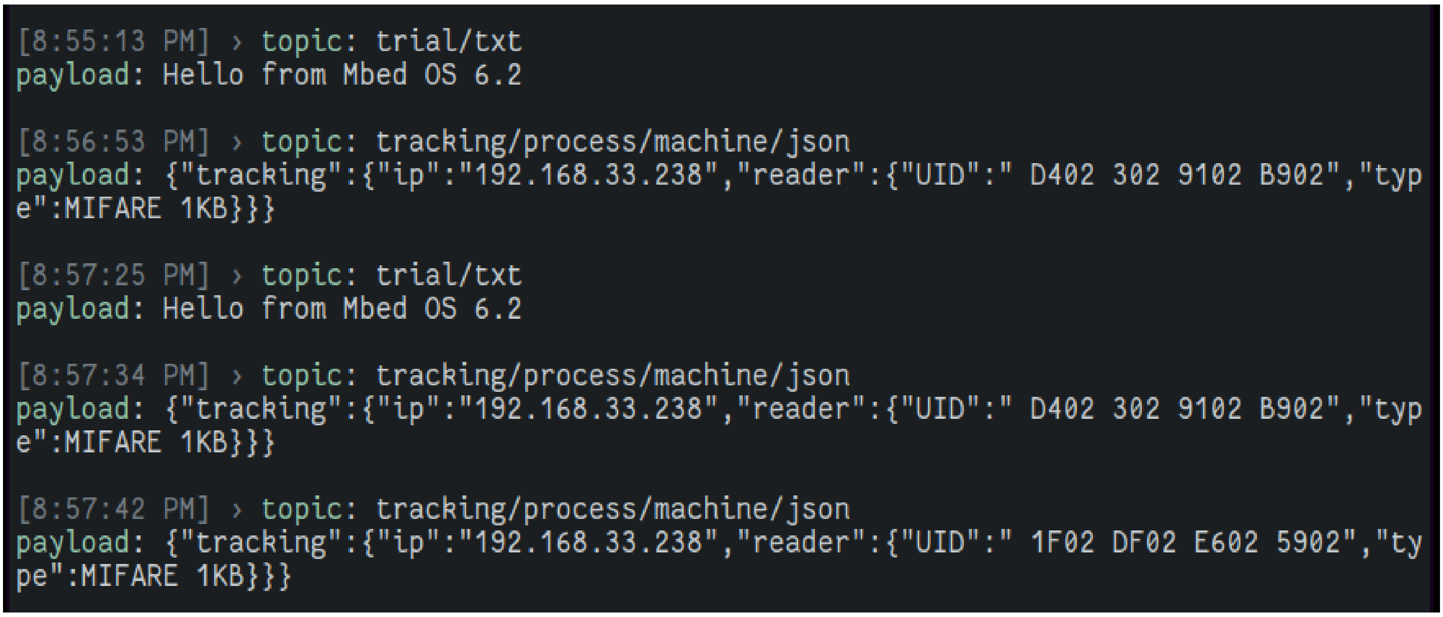
It works.

By checking the printf output using a serial console. It possible to follow up each step as expected.



```
Connecting wifi..
RF chip initailzed

Connection success
IP via DHCP: 192.168.33.238
MQTT broker 186.237.58.214:[
Connecting mqtt.33co.de ...
Publishing with payload length 22
Card UID: D4023029102B902
UID:  D402 302 9102 B902
PICC Type: MIFARE 1KB
Publishing with payload length 93
Card UID: 1F02DF02E6025902
UID:  1F02 DF02 E602 5902
PICC Type: MIFARE 1KB
Publishing with payload length 94
▯

CTRL-A Z for help | 9600 8N1 | NOR | Minicom 2.8 | VT102 | Offline | ttyUSB0
```

Also, by subscribing the MQTT topic it's possible to validate the entire solution.



```
[8:55:13 PM] › topic: trial/txt
payload: Hello from Mbed OS 6.2

[8:56:53 PM] › topic: tracking/process/machine/json
payload: {"tracking":{"ip":"192.168.33.238","reader":{"UID":" D402 302 9102 B902","typ
e":MIFARE 1KB}}}

[8:57:25 PM] › topic: trial/txt
payload: Hello from Mbed OS 6.2

[8:57:34 PM] › topic: tracking/process/machine/json
payload: {"tracking":{"ip":"192.168.33.238","reader":{"UID":" D402 302 9102 B902","typ
e":MIFARE 1KB}}}

[8:57:42 PM] › topic: tracking/process/machine/json
payload: {"tracking":{"ip":"192.168.33.238","reader":{"UID":" 1F02 DF02 E602 5902","ty
pe":MIFARE 1KB}}}
```

During the development phase, there was more output to check variable output and debug the code. The final code has s minimal console output.