```
#include <RMaker.h>
#include<string.h>
#include <WiFi.h>
#include <WiFiProv.h>
#include <SimpleTimer.h>
#include <EEPROM.h>
#include <MFRC522.h>
#include <SPI.h>

// SET DEFAULTS
#define DEFAULT_RELAY_MODE true

//BLE CREDENTIALS
const char *service_name = "PROV_RFID_techiesms";
const char *pop = "1234567";

//GPIO
static uint8_t gpio_reset = 0;
static uint8_t SS_PIN = 21;
static uint8_t RST_PIN = 5;
static uint8_t Internet_LED = 14;
static uint8_t Read_Mode_LED = 13;
static uint8_t RELAY = 33;
static uint8_t SWITCH = 32;
static uint8_t BUZZER = 26;

bool relay_state = false;
bool wifi_connected = 0;
bool add_button = false;
bool remove_button = false;
int SWITCH_STATE = HIGH;
int address;
bool buzzer_state = false;
bool buzz = false;

MFRC522 mfrc522(SS_PIN, RST_PIN);

SimpleTimer Timer;

//------------------------------------------
Declaring Device
static Device my_lock("RFID LOCK",
"custom.device.device");

void sysProvEvent(arduino_event_t *sys_event)
{
  switch (sys_event->event_id) {
    case ARDUINO_EVENT_PROV_START:
      Serial.println("CLEARING EEPROM...");
      for (int i = 0; i < 512; i++)
      {
        EEPROM.write(i, 0);
      }
      EEPROM.commit();
#if CONFIG_IDF_TARGET_ESP32
      Serial.printf("\nProvisioning Started with
name \"%s\" and PoP \"%s\" on BLE\n",
service_name, pop);
      printQR(service_name, pop, "ble");
#else
      Serial.printf("\nProvisioning Started with
name \"%s\" and PoP \"%s\" on SoftAP\n",
service_name, pop);
      printQR(service_name, pop, "softap");
#endif
      break;
    case ARDUINO_EVENT_WIFI_STA_CONNECTED:
      Serial.printf("\nConnected to Wi-Fi!\n");
      digitalWrite(Internet_LED, HIGH);
      //wifi_connected = 1;
      delay(500);
      break;
    case ARDUINO_EVENT_WIFI_STA_DISCONNECTED:
      Serial.println("\nDisconnected. Connecting
to the AP again... ");
      digitalWrite(Internet_LED, LOW);
      break;
    case ARDUINO_EVENT_PROV_CRED_RECV: {
        Serial.println("\nReceived Wi-Fi
credentials");
        Serial.print("\tSSID : ");
        Serial.println((const char *)
sys_event->event_info.prov_cred_recv.ssid);
        Serial.print("\tPassword : ");
        Serial.println((char const *)
sys_event->event_info.prov_cred_recv.password);
        break;
      case ARDUINO_EVENT_PROV_INIT:
        wifi_prov_mgr_disable_auto_stop(10000);
        break;
      case ARDUINO_EVENT_PROV_CRED_SUCCESS:
        Serial.println("Stopping Provisioning!!!
");
```

```
        wifi_prov_mgr_stop_provisioning();
        break;
      }
  }
}

void write_callback(Device *device, Param
*param, const param_val_t val, void *priv_data,
write_ctx_t *ctx)
{
  const char *device_name =
device->getDeviceName();
  Serial.println(device_name);
  const char *param_name = param->getParamName();

  if (strcmp(device_name, "RFID LOCK") == 0)
  {
    if (strcmp(param_name, "display") == 0) {

      Serial.printf("\n Access %s \n", val.val.
s);
      param->updateAndReport(val);
    }
    if (strcmp(param_name, "BUZZER") == 0)
    {
      Serial.printf("Buzzer ", val.val.b ? "ON"
: "OFF");
      buzzer_state = val.val.b;
      if (buzzer_state == true)
      {
        buzz = true;
        beep();
      }
      else
      {
        buzz = false;
      }
      param->updateAndReport(val);
    }
    if (strcmp(param_name, "DOOR OPEN") == 0)
    {
      Serial.printf("Door %s", val.val.b ?
"OPEN" : "CLOSED");
      relay_state = val.val.b;
      (relay_state == false) ?
digitalWrite(RELAY, HIGH) : digitalWrite(RELAY,
LOW);
      param->updateAndReport(val);
    }

    if (strcmp(param_name, "ADD RFID") == 0)
    {
      digitalWrite(Read_Mode_LED, HIGH);
      beep();
      Serial.printf("\nScan RFID\n");
      add_button = val.val.b;
      if (add_button == true)
      {
        while ( ! mfrc522.
PICC_IsNewCardPresent())
        {
          continue;
        }
        // Select one of the cards
        while ( ! mfrc522.PICC_ReadCardSerial())
        {
          continue;
        }

        Serial.print("UID tag :");
        String tid = "";
        byte let;
        for (byte i = 0; i < mfrc522.uid.size;
i++)
        {
          Serial.print(mfrc522.uid.uidByte[i] <
0x10 ? " 0" : " ");
          Serial.print(mfrc522.uid.uidByte[i],
HEX);
          tid.concat(String(mfrc522.uid.
uidByte[i] < 0x10 ? " 0" : " "));
          tid.concat(String(mfrc522.uid.
uidByte[i], HEX));
        }
        tid.toUpperCase();

        Serial.println(tid);

        String read = readStringFromEeprom(0);
        read.toUpperCase();

        if (read.indexOf(tid) != -1)
        {
          Serial.println("rfid available");
          Failure_buzzer();
        }
        else
        {
          writeStringTOEeprom(address, tid);
          Serial.println("RFID ADDED
SUCCESSFULLY");
          success_buzzer();
          add_switch_off();
        }
      }
      else
      {
        Failure_buzzer();
        add_switch_off();
      }
      digitalWrite(Read_Mode_LED, LOW);
      add_switch_off();
      delay(1000);

    }

    if (strcmp(param_name, "REMOVE RFID") == 0)
    {
      Serial.printf("\nScan RFID\n");
      beep();
      digitalWrite(Read_Mode_LED, HIGH);
      remove_button = val.val.b;
      if (remove_button == true)
      {
        while ( ! mfrc522.
PICC_IsNewCardPresent())
        {
          continue;
        }
        // Select one of the cards
        while ( ! mfrc522.PICC_ReadCardSerial())
        {
          continue;
        }

        Serial.print("UID tag :");
        String tid = "";
        byte let;
        for (byte i = 0; i < mfrc522.uid.size;
i++)
        {
          Serial.print(mfrc522.uid.uidByte[i] <
0x10 ? " 0" : " ");
          Serial.print(mfrc522.uid.uidByte[i],
HEX);
          tid.concat(String(mfrc522.uid.
uidByte[i] < 0x10 ? " 0" : " "));
          tid.concat(String(mfrc522.uid.
uidByte[i], HEX));
        }
        tid.toUpperCase();

        Serial.print("TID - "); Serial.
println(tid);
```

```
        String read = readStringFromEeprom(0);
        Serial.print("String Length Before
Removal - "); Serial.println(read.length());
        read.toUpperCase();

        if (read.indexOf(tid) == -1)
        {
          Serial.println("rfid not available");
          Failure_buzzer();
        }
        else
        {
          read.remove(read.indexOf(tid), 12);
          Serial.println("CLEARING EEPROM...");
          for (int i = 0; i < 512; i++)
          {
            EEPROM.write(i, 0);
          }
          EEPROM.commit();
          writeStringTOEeprom(0, read);

          read = readStringFromEeprom(0);
          Serial.print("String Length After
Removal - "); Serial.println(read.length());
          Serial.println(read);
          Serial.println("RFID REMOVED
SUCCESSFULLY");
          success_buzzer();
        }
      }
      else
      {
        remove_switch_off();
        Failure_buzzer();
      }
      digitalWrite(Read_Mode_LED, LOW);
      remove_switch_off();
      delay(1000);
    }
  }
}

void setup() {
  Serial.begin(115200);

  if (!EEPROM.begin(512))
  {
    Serial.println("Failed to initialize 
EEPROM");
    delay(1000000);
  }

  SPI.begin();
  mfrc522.PCD_Init();

  pinMode(Read_Mode_LED, OUTPUT);
  pinMode(Internet_LED, OUTPUT);

  pinMode(gpio_reset , INPUT);
  pinMode(RELAY, OUTPUT);
  pinMode(SWITCH, INPUT);
  pinMode(BUZZER, OUTPUT);
  digitalWrite(RELAY, DEFAULT_RELAY_MODE);

  Serial.println("Put your card to the reader...
");
  Serial.println();



  //------------------------------------------------
Declaring Node
  Node my_node;
  my_node = RMaker.initNode("Techiesms");



  //------------------------------------------------
Declaring Parameters
  my_lock.addNameParam();
  Param disp("display", "custom.param.display",
value("Welcome to techiesms"), PROP_FLAG_READ);
  disp.addUIType(ESP_RMAKER_UI_TEXT);
  my_lock.addParam(disp);

  Param open_switch("DOOR OPEN", "custom.param.
power", value(relay_state), PROP_FLAG_READ |
PROP_FLAG_WRITE);
  open_switch.addUIType(ESP_RMAKER_UI_TOGGLE);
  my_lock.addParam(open_switch);

  Param add_switch("ADD RFID", "custom.param.
power", value(add_button), PROP_FLAG_READ |
PROP_FLAG_WRITE);
  add_switch.addUIType(ESP_RMAKER_UI_TOGGLE);
  my_lock.addParam(add_switch);

  Param remove_switch("REMOVE RFID",
"custom.param.power", value(remove_button),
PROP_FLAG_READ | PROP_FLAG_WRITE);
  remove_switch.addUIType(ESP_RMAKER_UI_TOGGLE);
  my_lock.addParam(remove_switch);

  Param buzz_switch("BUZZER", "custom.param.
power", value(buzzer_state), PROP_FLAG_READ |
PROP_FLAG_WRITE);
  buzz_switch.addUIType(ESP_RMAKER_UI_TOGGLE);
  my_lock.addParam(buzz_switch);

  my_lock.addCb(write_callback);

  my_node.addDevice(my_lock);

  //DEFAULTS
  my_lock.updateAndReportParam("DOOR OPEN",
relay_state);
  my_lock.updateAndReportParam("ADD RFID",
add_button);
  my_lock.updateAndReportParam("REMOVE RFID",
remove_button);
  my_lock.updateAndReportParam("BUZZER",
buzzer_state);

  RMaker.enableOTA(OTA_USING_PARAMS);
  RMaker.enableTZService();
  RMaker.enableSchedule();
  Serial.printf("\nStarting ESP-RainMaker\n");
  RMaker.start();

  WiFi.onEvent(sysProvEvent);
  WiFiProv.beginProvision(WIFI_PROV_SCHEME_BLE,
WIFI_PROV_SCHEME_HANDLER_FREE_BTDM,
WIFI_PROV_SECURITY_1, pop, service_name);
  delay(1000);
  address = EEPROM.read(500);
  Serial.println(address);
}

void loop()
{
  //--------------------------------- Exit
Switch

  SWITCH_STATE = digitalRead(SWITCH);
  if (SWITCH_STATE == LOW)
  {
    authorized_access_offline();
  }

  String s1 = getValueFromRfid();
  char str[s1.length() + 1];

  for (int i = 0; i < s1.length(); i++) {
    str[i] = s1[i];
  }
  my_lock.updateAndReportParam("display", str);

  if (strstr(str, "Access Denied"))
  {
    delay(2000);
  }

  //RESET USING GPIO 0
  if (digitalRead(gpio_reset) == LOW) { //Push
button pressed
    Serial.printf("Reset Button Pressed!\n");
    // Key debounce handling
    delay(100);
    int startTime = millis();
    while (digitalRead(gpio_reset) == LOW)
delay(50);
    int endTime = millis();

    if ((endTime - startTime) > 5000) {
      // If key pressed for more than 5 sec,
reset all
      Serial.printf("Reset to factory.\n");
      RMakerFactoryReset(2);
    }
  }
  delay(100);
}

String getValueFromRfid()
{
  // Look for new cards
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return "";
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return "";
  }
  //Show UID on serial monitor
  Serial.print("UID tag :");
  String content = "";
  byte letter;
  String val = "";
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ?
" 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i]
< 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.
uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();

  val = compareUID(content);
  return val;
}
```

```
void add_switch_off(void)
{
  add_button = false;
  my_lock.updateAndReportParam("ADD RFID",
add_button);
  Serial.println("Button off");
}

void remove_switch_off(void)
{
  remove_button = false;
  my_lock.updateAndReportParam("REMOVE RFID",
remove_button);
  Serial.println("Button off");
}


String authorized_access(void)
{
  Serial.println("Authorized access");
  Serial.println();
  my_lock.updateAndReportParam("display",
"Access Authorized");
  beep();
  digitalWrite(RELAY, LOW);
  delay(5000);
  digitalWrite(RELAY, HIGH);

  return "Access Authorized";
```

```
}

void writeStringTOEeprom(int add, String str)
{
  int loc = 0;
  int len = str.length();

  for (int i = 0; i < len; i++)
  {
    loc = add  + 1 + i;
    EEPROM.write(add + 1 + i, str[i]);
    Serial.print("Wrote: ");
    Serial.println(str[i]);
    Serial.print("Location - "); Serial.
println(loc);
    Serial.println("");
  }
  address = loc;
  Serial.println(address);
  Serial.println(str);
  EEPROM.write(500, address);
  EEPROM.commit();
  delay(2000);
}

String readStringFromEeprom(int add)
{
  int len = address;
  String MyString;

  for (int i = 0; i < len; i++) {
    MyString += char(EEPROM.read(add + 1 + i));

  }
  return String(MyString);
}

String compareUID(String str)
{
  String val = "";
  String read = readStringFromEeprom(0);

  if (read.indexOf(str) != -1)
  {
    val = authorized_access();
    return val;
  }
  else   {
    Serial.println(" Access denied");
    digitalWrite(RELAY, HIGH);
    Failure_buzzer();
    return "Access Denied";
  }
}

void authorized_access_offline()
{
  Serial.println("Authorized access");
  digitalWrite(RELAY, LOW);
  delay(5000);
  digitalWrite(RELAY, HIGH);
}
```

```
//---------------------------- Different Buzzer
Patterns

void success_buzzer()
{
  if (buzz == true)
  {
    digitalWrite(BUZZER, HIGH);
    delay(2000);
    digitalWrite(BUZZER, LOW);
  }
}

void Failure_buzzer()
{
  if (buzz == true)
  {
    for (int i = 0; i < 3; i++)
    {
      digitalWrite(BUZZER, HIGH);
      delay(100);
      digitalWrite(BUZZER, LOW);
      delay(50);
    }
  }
}

void beep()
{
  if (buzz == true)
  {
    digitalWrite(BUZZER, HIGH);
    delay(100);
    digitalWrite(BUZZER, LOW);
  }
}
```

```
/*
void setup() {
  pinDefine(12);
  pinDefine(11);
  pinDefine(10);
  pinDefine(9);
}

void loop() {
  ledDefine(12, 1000, 1500);
  ledDefine(11, 1000, 1500);
  ledDefine(10, 1000, 1500);
  ledDefine(9, 1000, 1500);
}

void ledDefine(int led, int onTime, int offTime){
  digitalWrite(led, HIGH);
  delay(onTime);
  digitalWrite(led, LOW);
  delay(offTime);
}
void pinDefine(int pin){
  pinMode(pin, OUTPUT);
}

*/
```