**BÀI LAB MÔN AN TOÀN HỆ THỐNG NHÚNG**

Họ và tên : Nguyễn Thị Huyền Thương

Mã SV: DT040150

Lớp : L01

**Yêu cầu lab:**

Cài đặt thuật toán mật mã AES-128 trên kit ESP32. Nhận bản rõ từ PC, mã hóa bởi AES-128 và gửi trả lại bản mã cho PC. Kiểm tra tính đúng đắn mã hóa giải mã.

**Mục đích:**

Bảo mật dữ liệu thông qua việc sử dụng thuật toán mã hóa AES-128.

**Mô tả lab:**

1. **Phần cứng**

- sử dụng ESP32, nạp code qua Arduino IDE.

- mã hóa và giải mã AES cơ bản bằng cách sử dụng thư viện mbedTLS.

**Mô tả chương trình**

1. Thư viện và các hàm:

Chương trình bao gồm thư viện mbedtls/aes.h để sử dụng các chức năng mã hóa và giải mã AES.

1. Định nghĩa hai hàm:

- encrypt(char \*plainText, char \*key, unsigned char \*outputBuffer): Mã hóa văn bản thuần bằng chế độ AES-ECB.

- decrypt(unsigned char \*cipherText, char \*key, unsigned char \*outputBuffer): Giải mã văn bản đã mã hóa bằng chế độ AES-ECB.

3. Hàm mã hóa và giải mã AES:

- Hàm mã hóa:

+ Khởi tạo ngữ cảnh AES.

+ Thiết lập khóa mã hóa.

+ Mã hóa văn bản thuần.

+ Giải phóng ngữ cảnh AES.

- Hàm giải mã:

+ Khởi tạo ngữ cảnh AES.

+ Thiết lập khóa giải mã.

+ Giải mã văn bản đã mã hóa.

+ Giải phóng ngữ cảnh AES.

- Hàm setup:

+ Khởi tạo giao tiếp nối tiếp ở tốc độ baud 115200.

+ Chờ cổng nối tiếp kết nối.

- Hàm loop:

+ Liên tục kiểm tra xem có dữ liệu nối tiếp nào sẵn có không.

+ Đọc tối đa 16 ký tự từ đầu vào nối tiếp vào bộ đệm.

+ Kết thúc chuỗi bằng ký tự null.

+ Đảm bảo độ dài đầu vào chính xác là 16 ký tự (kích thước khối AES).

+ Tiến hành mã hóa và giải mã văn bản đầu vào.

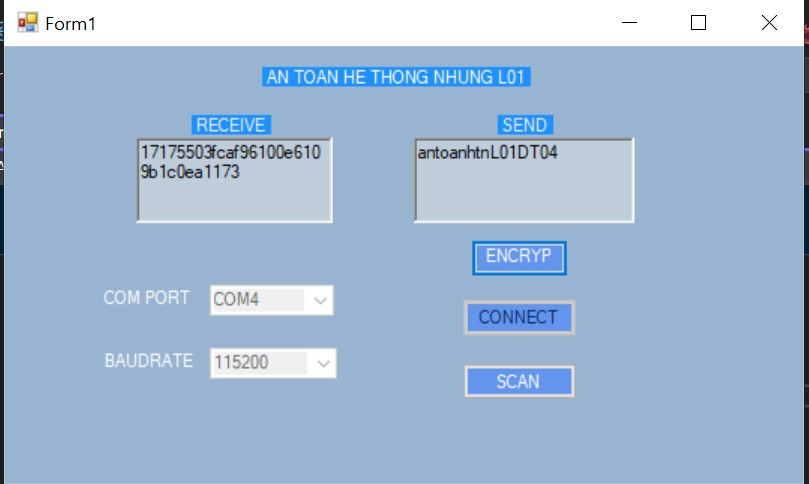
+ In dữ liệu đã mã hóa dưới dạng giá trị thập lục phân.

**2. Chương trình Tools ( sử dụng C#)**

Thiết lập kết nối với ESP32 qua cổng Serial: Chương trình C# cho phép người dùng chọn cổng COM và tốc độ baud để thiết lập kết nối với ESP32 thông qua cổng Serial.

Nhập và gửi văn bản đến ESP32 để mã hóa: Người dùng có thể nhập văn bản vào một ô văn bản trên giao diện và gửi nó đến ESP32 qua cổng Serial để mã hóa với khóa key đã được thiết lập sẵn.

**3. Thực thi lab**

****

Đầu tiên người dùng sẽ nhấn nút SCAN => chọn cổng COM, tốc độ baudrate.

Sau đó chọn CONNECT.

Sau khi kết nối thành công thì phần chọn COM PORT và BAUDRATE sẽ mờ.

Sau đó người dùng nhập dữ liệu cần mã hóa vào ô SEND và chọn ENCRYPT.

Dữ liệu sau khi mã hóa được hiển thị tại ô RECEIVE.

**4. Phụ lục**

**Chương trình Firmware**

|  |
| --- |
| #include "mbedtls/aes.h"  void encrypt(char \* plainText, char \* key, unsigned char \* outputBuffer) {      mbedtls\_aes\_context aes;      mbedtls\_aes\_init(&aes);      mbedtls\_aes\_setkey\_enc(&aes, (const unsigned char\*) key, strlen(key) \* 8);      mbedtls\_aes\_crypt\_ecb(&aes, MBEDTLS\_AES\_ENCRYPT, (const unsigned char\*)plainText, outputBuffer);      mbedtls\_aes\_free(&aes);  }  void decrypt(unsigned char \* cipherText, char \* key, unsigned char \* outputBuffer) {      mbedtls\_aes\_context aes;      mbedtls\_aes\_init(&aes);      mbedtls\_aes\_setkey\_dec(&aes, (const unsigned char\*) key, strlen(key) \* 8);      mbedtls\_aes\_crypt\_ecb(&aes, MBEDTLS\_AES\_DECRYPT, (const unsigned char\*)cipherText, outputBuffer);      mbedtls\_aes\_free(&aes);  }  void setup() {      Serial.begin(115200);      while (!Serial) {          ;      }    }  void loop() {      if (Serial.available() > 0) {          char plainText[17] = {0}; // Buffer for plain text input          char key[] = "abcdefghijklmnop"; // 16-byte key          unsigned char cipherTextOutput[16];          unsigned char decipheredTextOutput[16];          int length = Serial.readBytes(plainText, 16); // Read up to 16 characters          plainText[length] = '\0'; // Null-terminate the string          // Ensure the input length is 16 characters          if (length != 16) {              return;          }          encrypt(plainText, key, cipherTextOutput);          decrypt(cipherTextOutput, key, decipheredTextOutput);            for (int i = 0; i < 16; i++) {              char str[3];              sprintf(str, "%02x", (int)cipherTextOutput[i]);              Serial.print(str);          }          for (int i = 0; i < 16; i++) {            }        }  } |

*Chương trình Tools*

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.IO.Ports;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using static System.Windows.Forms.VisualStyles.VisualStyleElement;  namespace ATHTN0011  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void serialPort\_DataReceived(object sender, EventArgs e)  {  string read = serialPort1.ReadExisting();  inTextBox.Text = inTextBox.Text + read;  }  private void btnScan\_Click(object sender, EventArgs e)  {  comPort.Text = "";  comPort.Items.Clear();  String[] ports = SerialPort.GetPortNames();  comPort.Items.AddRange(ports);  }  private void btnConnect\_Click(object sender, EventArgs e)  {  if (ConnectPort() == true)  {  btnConnect.Enabled = false;  comPort.Enabled = false;  baudrate.Enabled = false;  }  }  public bool ConnectPort()  {  try  {  if (comPort.Text != "" || baudrate.Text != "")  {  serialPort1.PortName = comPort.Text;  serialPort1.Open();  serialPort1.BaudRate = Int32.Parse(baudrate.Text);  return true;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK);  }  return false;  }  private void outTextBox\_TextChanged(object sender, EventArgs e)  {  }  private void inTextBox\_TextChanged(object sender, EventArgs e)  {  }  private void btnSend\_Click(object sender, EventArgs e)  {  try  {  if (!(serialPort1.IsOpen))  {  serialPort1.Open();  }  serialPort1.Write(outTextBox.Text);  // outTextBox.Text = "";  }  catch (Exception ex)  {  MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK);  }    /\*try  {  if (!(serialPort1.IsOpen))  {  serialPort1.Open();  }  serialPort1.Write(outTextBox.Text);  outTextBox.Text = "";  }  catch (Exception ex)  {  MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK);  }\*/  }  private void serialPort1\_DataReceived(object sender, SerialDataReceivedEventArgs e)  {  this.Invoke(new EventHandler(serialPort\_DataReceived));  }  private void baudrate\_SelectedIndexChanged(object sender, EventArgs e)  {  }  }  } |