Republic of Iraq
Ministry of Higher Education and
Scientific Research
University of Technology
Control and Systems Engineering Department

Secure Mechanism applied to communication system

Assistant Dr. Ekhlas K.Hamza
Supervisor
Control And Systems Department,
University of Technology –Iraq
Baghdad

Marwan A. Hussein
Student
Control And Systems Department,
University of Technology –Iraq
Baghdad

Outline

- Background
- Research Problem Statement.
- Research objectives.
- Research Methodology.
- Next Steps.

Background

Research Objective

Research Environment

Research Problem Statement

Design and Implementation of Communication system (IoT) Secure Technology

based on C Language and NodeMCU

Research Tool

Communication

The communication system is a system model that describes a communication exchange between two stations, transmitter, and receiver. Signals or information passes from source to destination through a channel. Based on physical infrastructure there are two types of communication systems:

Line communication systems

Radio communication systems

Secure Communication

secure communication is when two entities are communicating and do not want a third party to listen in. For that they need to communicate in a way not susceptible to eavesdropping or interception.

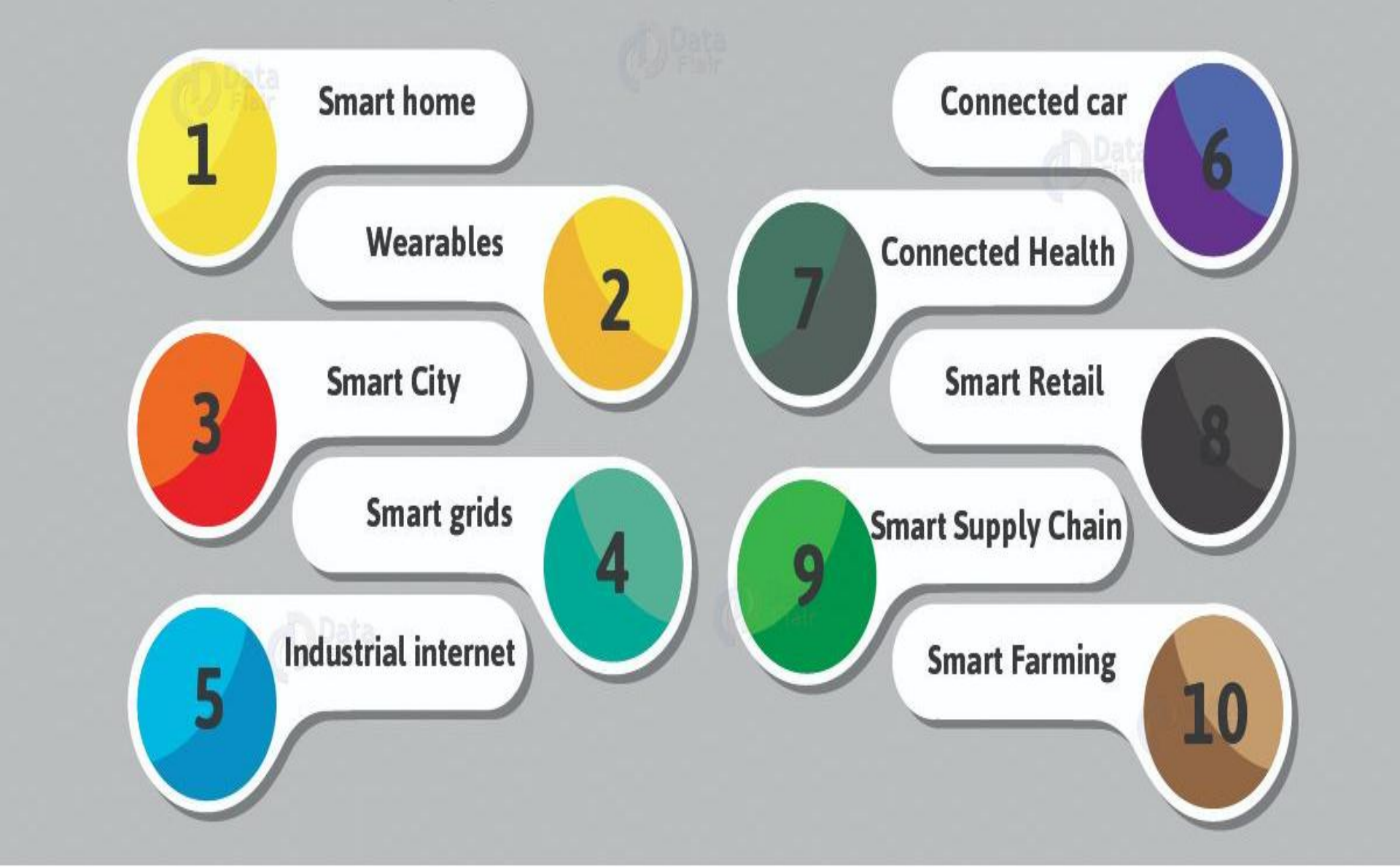
many communications taking place over long distance and mediated by technology, and increasing awareness of the importance of interception issues, technology and its compromise are at the heart of this debate

IOT

INTERNET OF THINGS

Internet of Things (IoT)

The Internet of Things is a term that has emerged recently, referring to the new generation of the Internet (the network) that allows understanding between devices interconnected with each other (via the Internet Protocol). These devices include tools, sensors, and various artificial intelligence tools, among others. This definition goes beyond the traditional concept that people communicate with computers and smartphones over a single global network and through the well-known traditional Internet Protocol. What distinguishes the Internet of Things is that it allows a person to be free from a place, meaning that a person can control the tools without the need to be in a specific place to deal with a specific device



IoT Applications and Use Cases

The Problem

Problem description

- Lack of security and privacy protection for existing IoT systems
- Heterogeneity
- User awareness
- Position of defense

Objective

Research Objectives

First

- Design a secure end-to-end Two-Factor Authentication Protocol (TFA) system

Second

- Implementing a system on two NodeMCU Devices

Third

- Addition of optimization to the system

Authors names	Year	Paper name	Description
Aman, Muhammad Naveed Basheer, Mohamed Haroon Sikdar, Biplab	2018	Two-Factor Authentication for IoT with Location Information	Proposes a two-factor authentication protocol using physically unclonable functions and the characteristics of the wireless signal from an IoT device
Liu, Zhenhua Guo, Changbo Wang, Baocang	2020	A Physically Secure, Lightweight Three-Factor and Anonymous User Authentication Protocol for IoT	The proposed protocol can provide the physical security through physically unclonable function (PUF), require no additional phase to update challenge-response pairs (CRPs), and store a single CRP for each sensor.

RELATED WORK

Research Methodology

First Stage

- Identify the basic elements of communication client and server
- Provide NodeMCU
- Install a program Arduino Software (IDE)
- Installing Esp8266 and adding libraries to Arduino Software (IDE)

Second Stage

- Using C language to implement a prototype of the TFA
- Implementation two-factor authentication protocol NodeMCU

Three stage

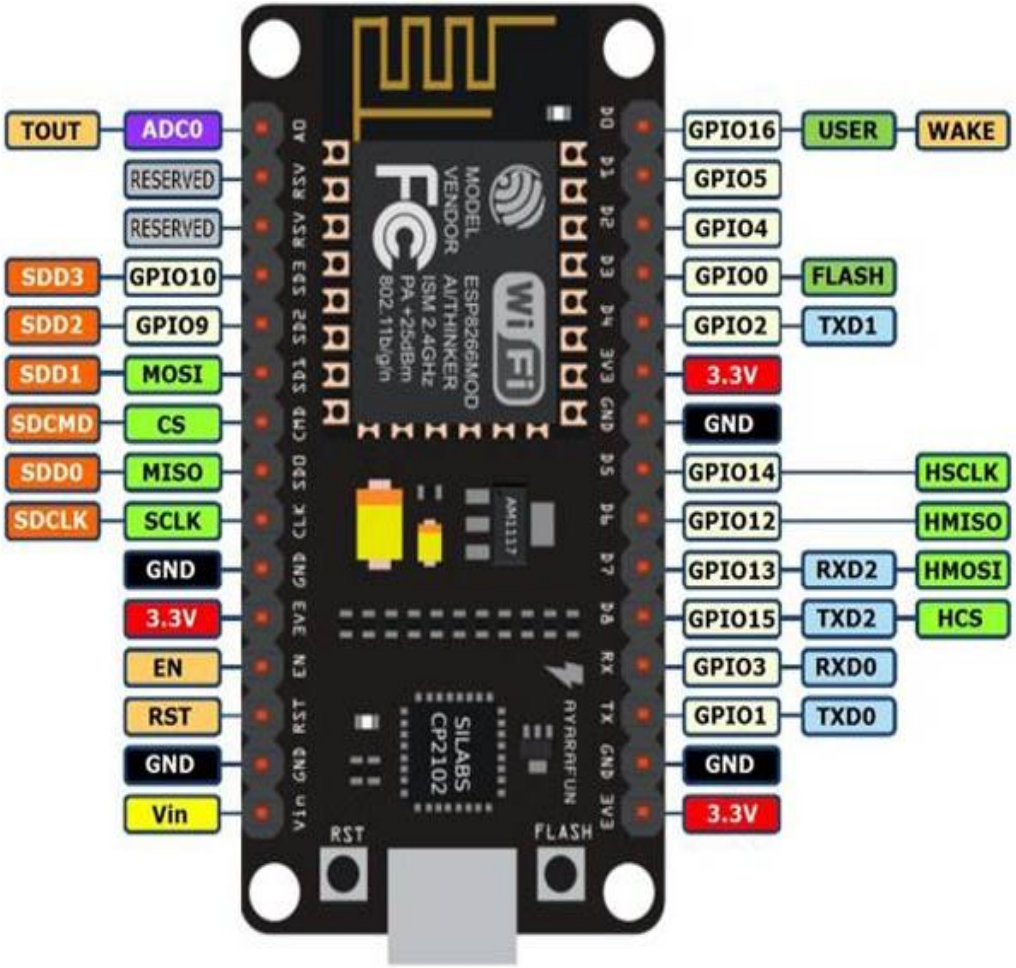
- Addition of optimizatio n to the system

NodeMCU ESP8266

NodeMCU is an open source firmware for which open source prototyping board designs are available.

The name "NodeMCU" combines "node" and "MCU" (micro-controller unit).

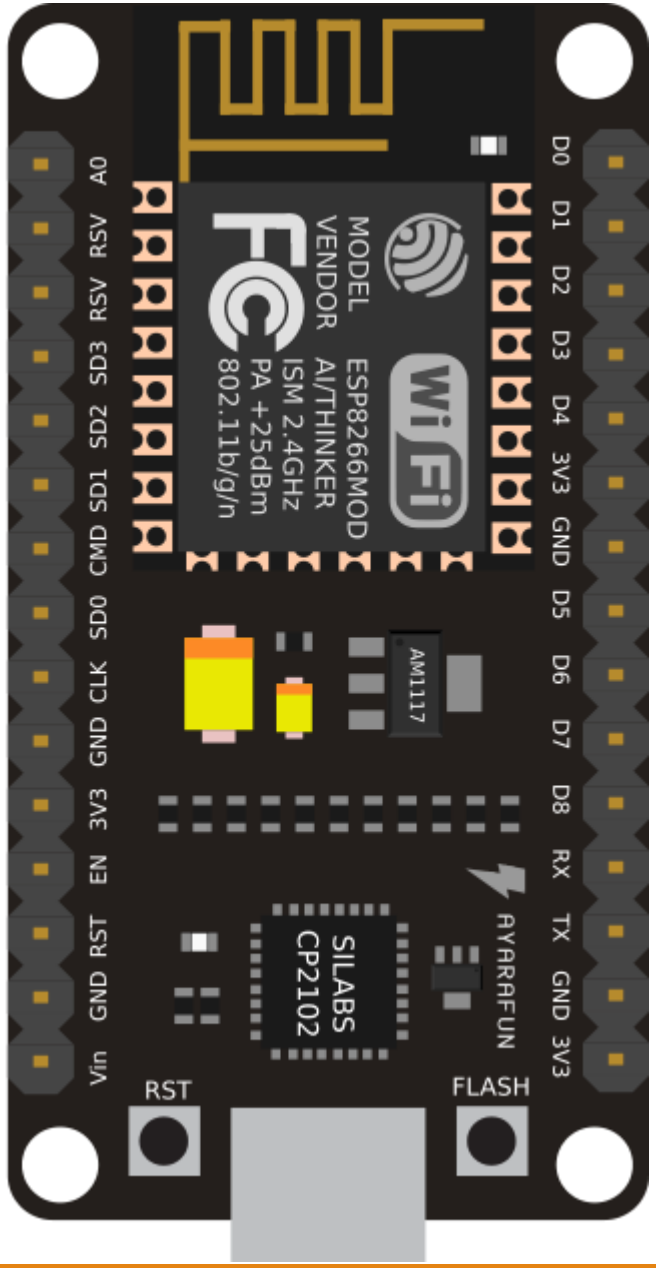
The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits.



NodeMCU ESP8266

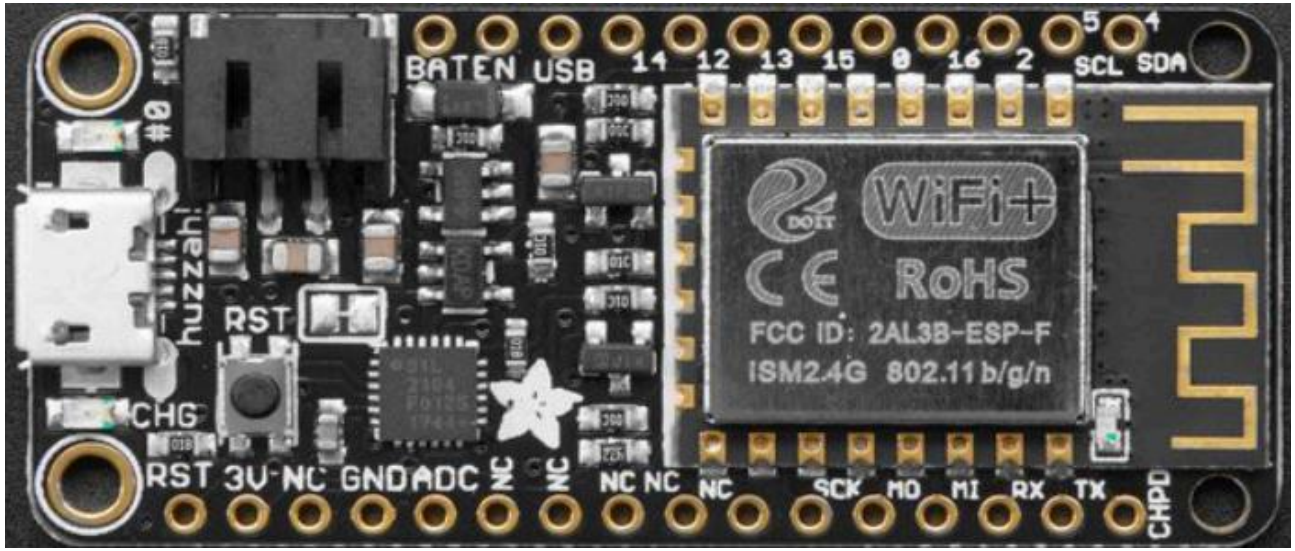
FEATURES

- Wi-Fi Module – ESP-12E module similar to ESP-12 module but with 6 extra GPIOs.
- USB – micro USB port for power, programming and debugging
- Headers – 2x 2.54mm 15-pin header with access to GPIOs, SPI, UART, ADC, and power pinsMisc – Reset and Flash buttons
- Power – 5V via micro USB port
- Dimensions – 49 x 24.5 x 13mm



Connect Things EASY

The NodeMCU is a development board featuring the popular ESP8266 WiFi chip. As it turns out, you can program the ESP8266 just like any other microcontroller. Its obvious advantage over the Arduino or PIC is that it can readily connect to the Internet via WiFi. However, the ESP8266 breakout board has limited pins although the chip itself has a lot of output ports. The NodeMCU solves this problem by featuring 10 GPIO pins each capable of using PWM, I2C and 1 –wire interface.

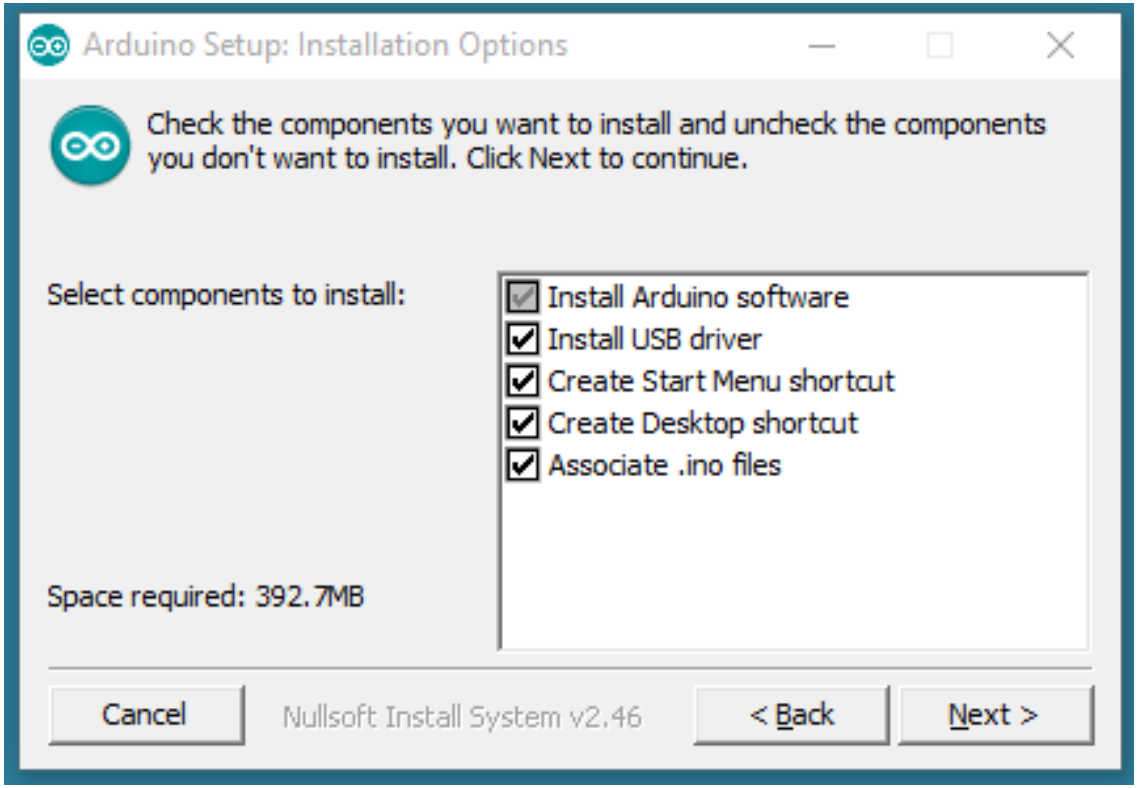


Install the Arduino Software (IDE) on Windows PCs

1. Download the Arduino Software (IDE) from the URL
   https://www.arduino.cc/en/software

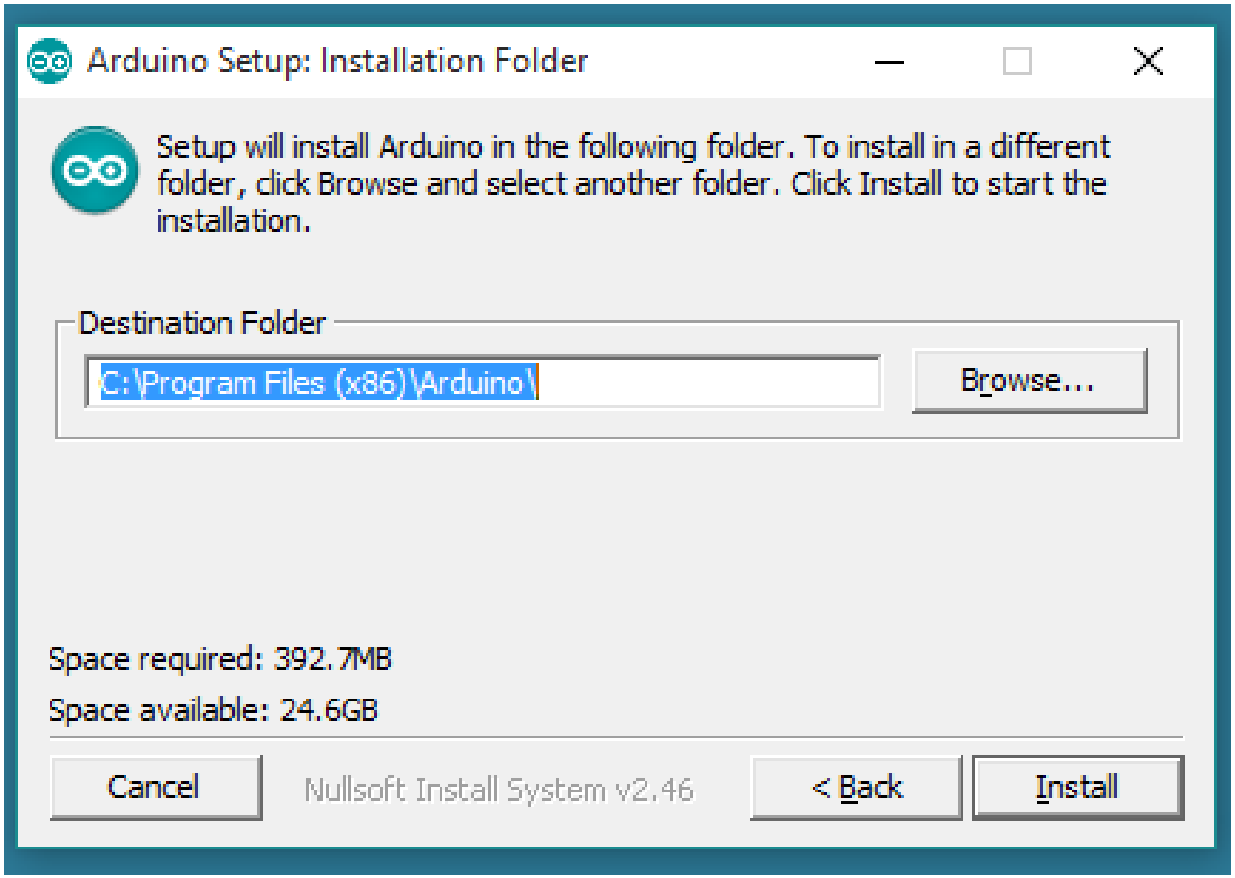
Install the Arduino Software (IDE) on Windows PCs

2. Choose the components .



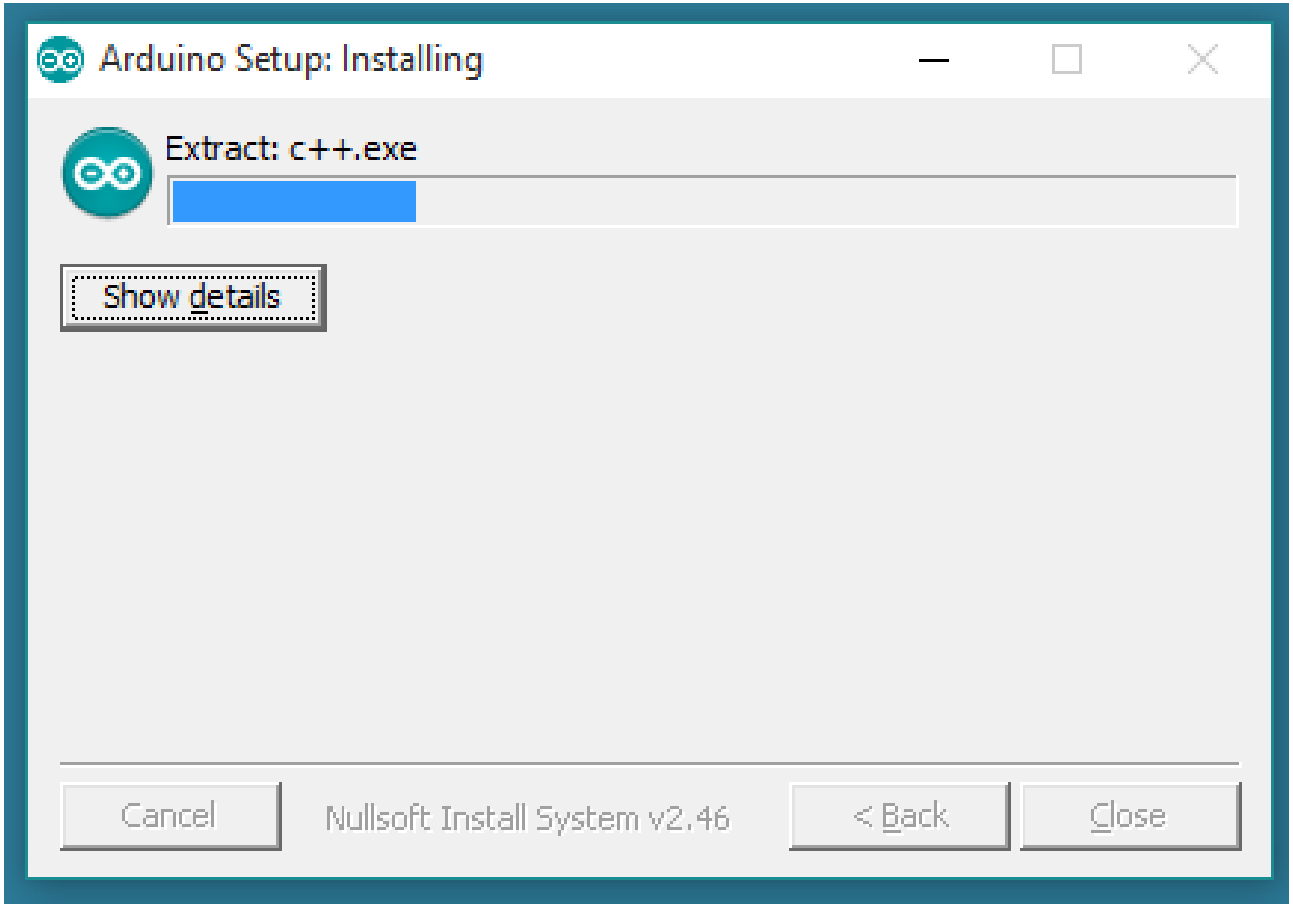
Install the Arduino Software (IDE) on Windows PCs

3. Choose the installation directory (we suggest to keep the default one).



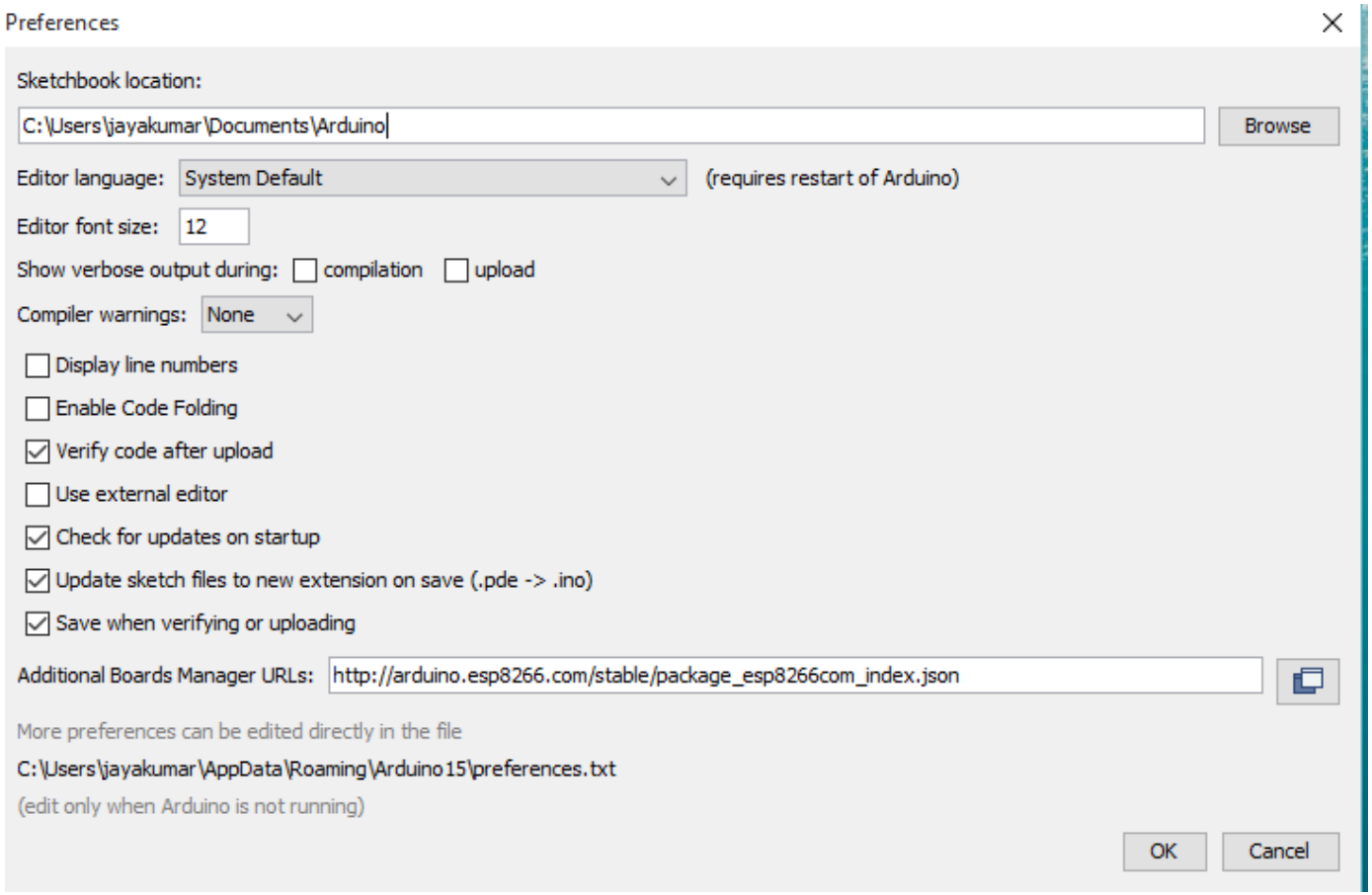
Install the Arduino Software (IDE) on Windows PCs

4. The process will extract and install all the required files to execute properly the Arduino Software (IDE).

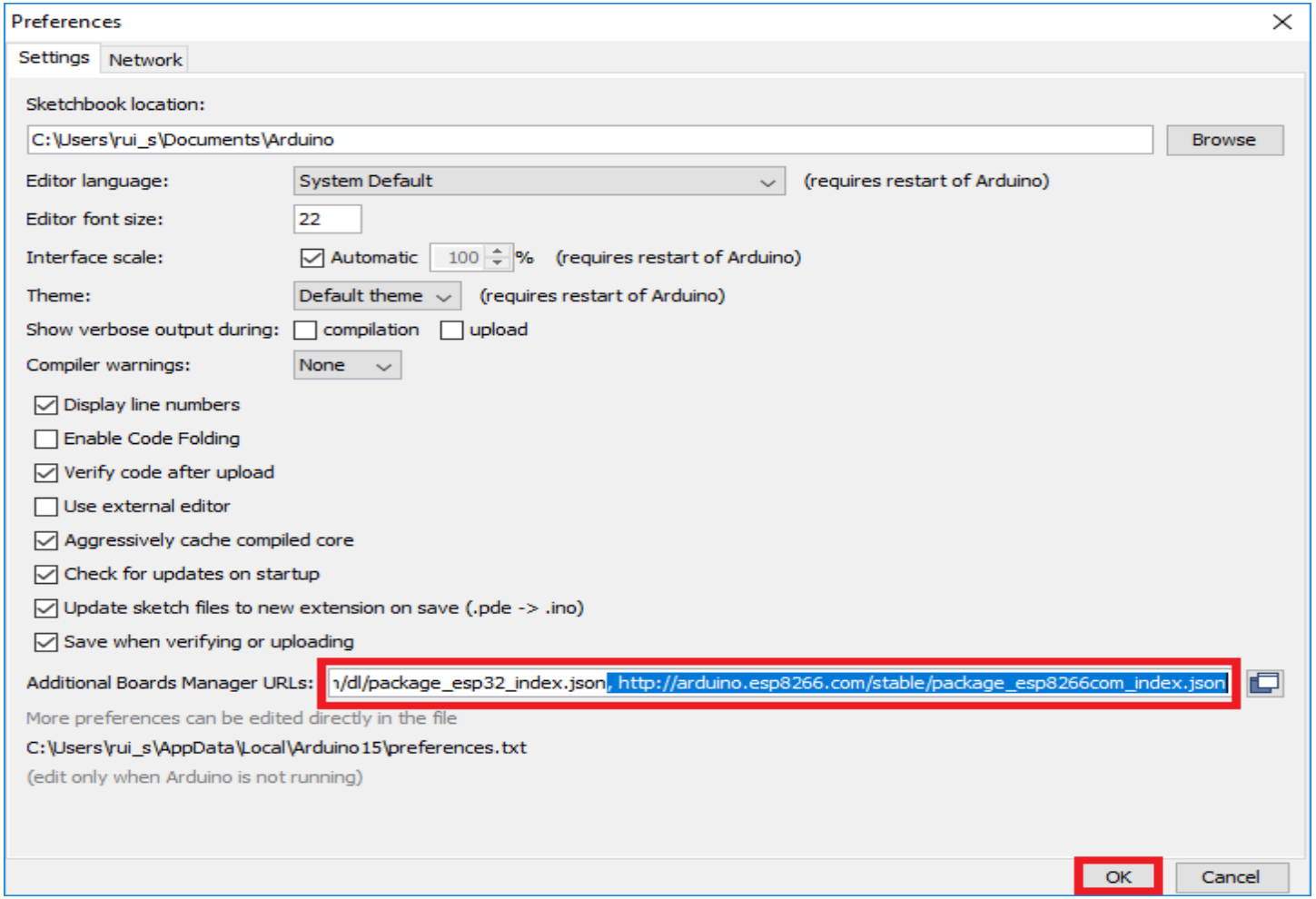


Firstly open the Arduino IDE

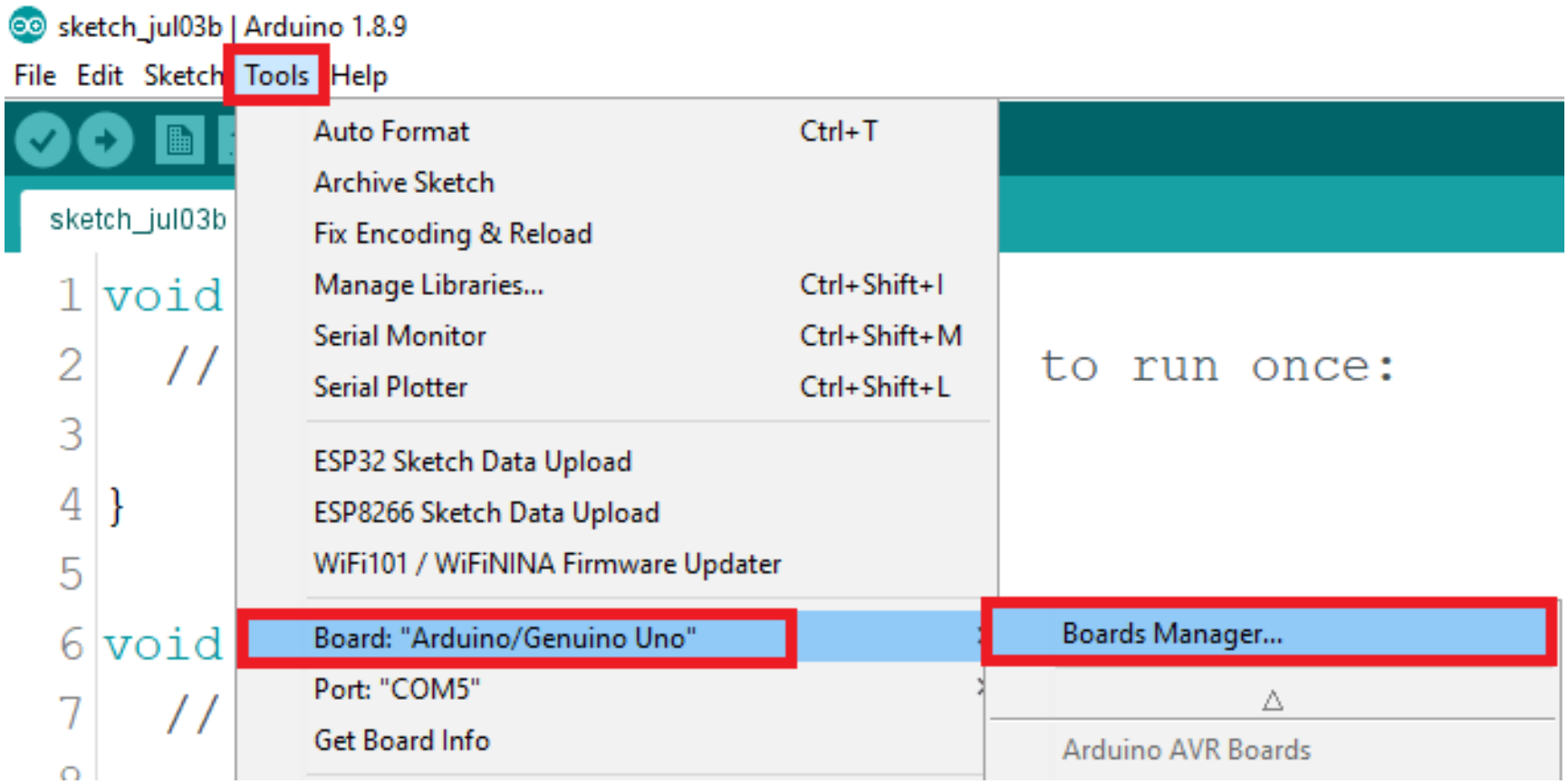
Go to files and click on the preference in the Arduino IDE



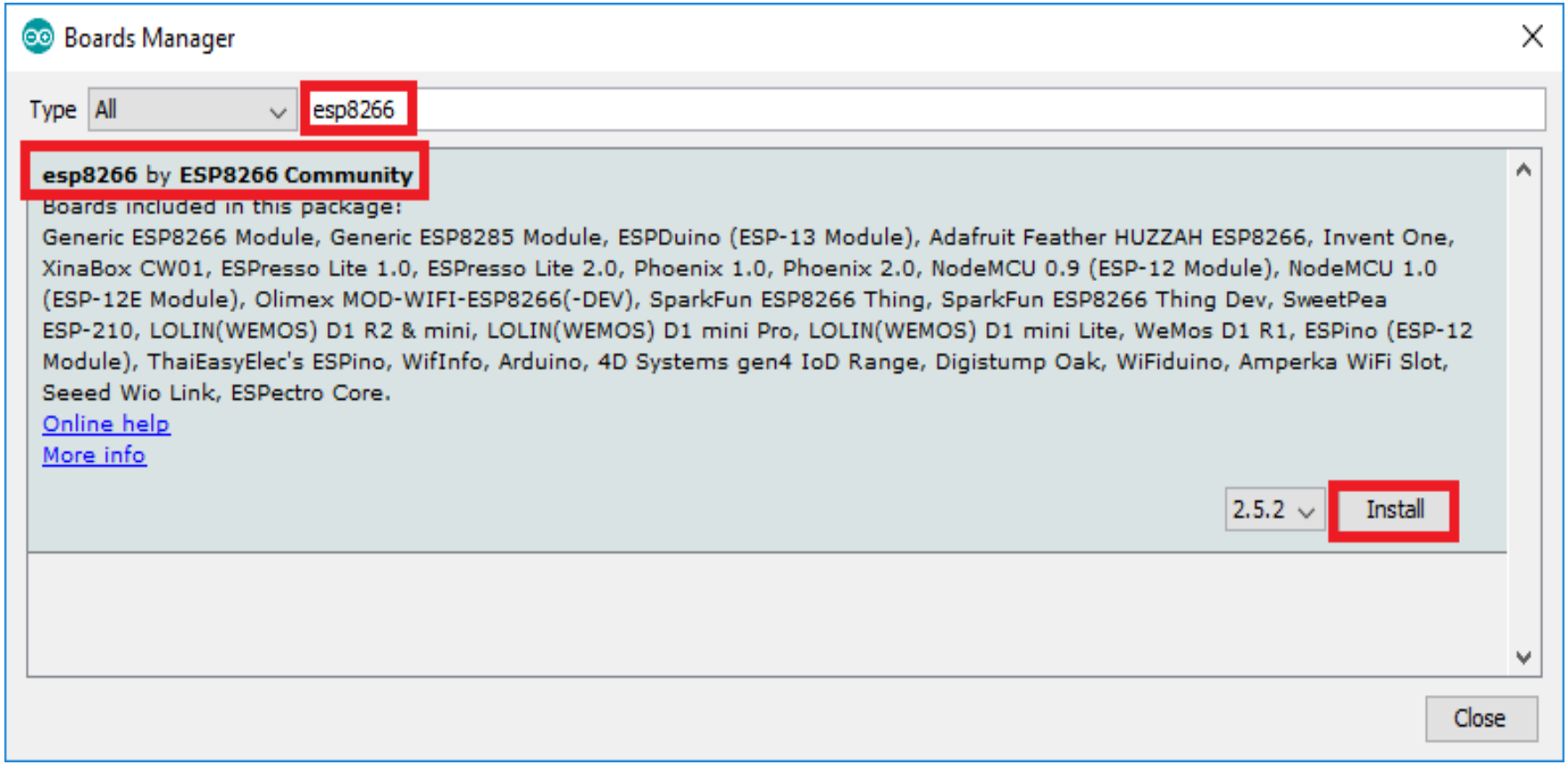
2.Enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into the “Additional Boards Manager URLs” field as shown in the figure below. Then, click the “OK” button:



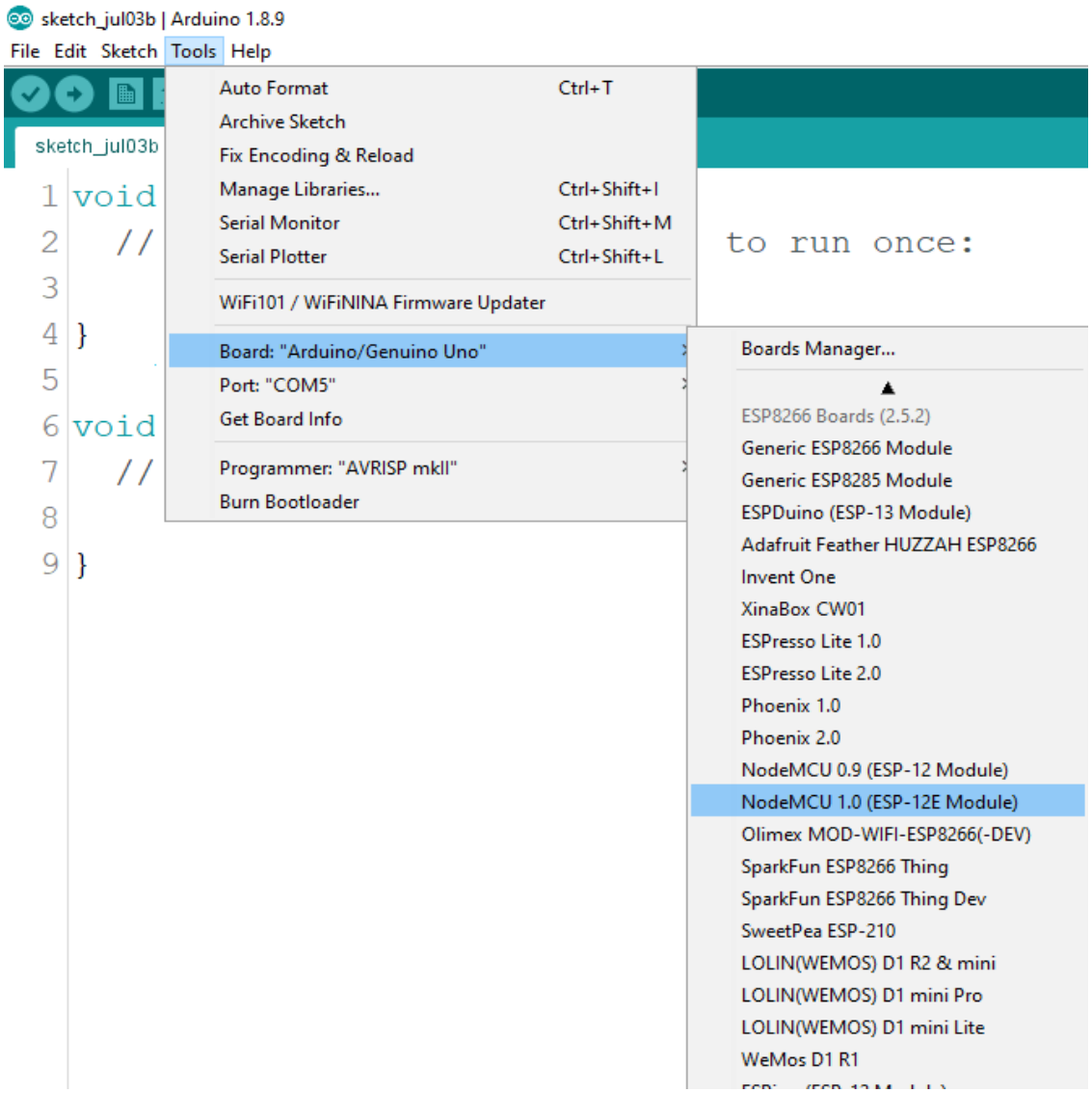
3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**



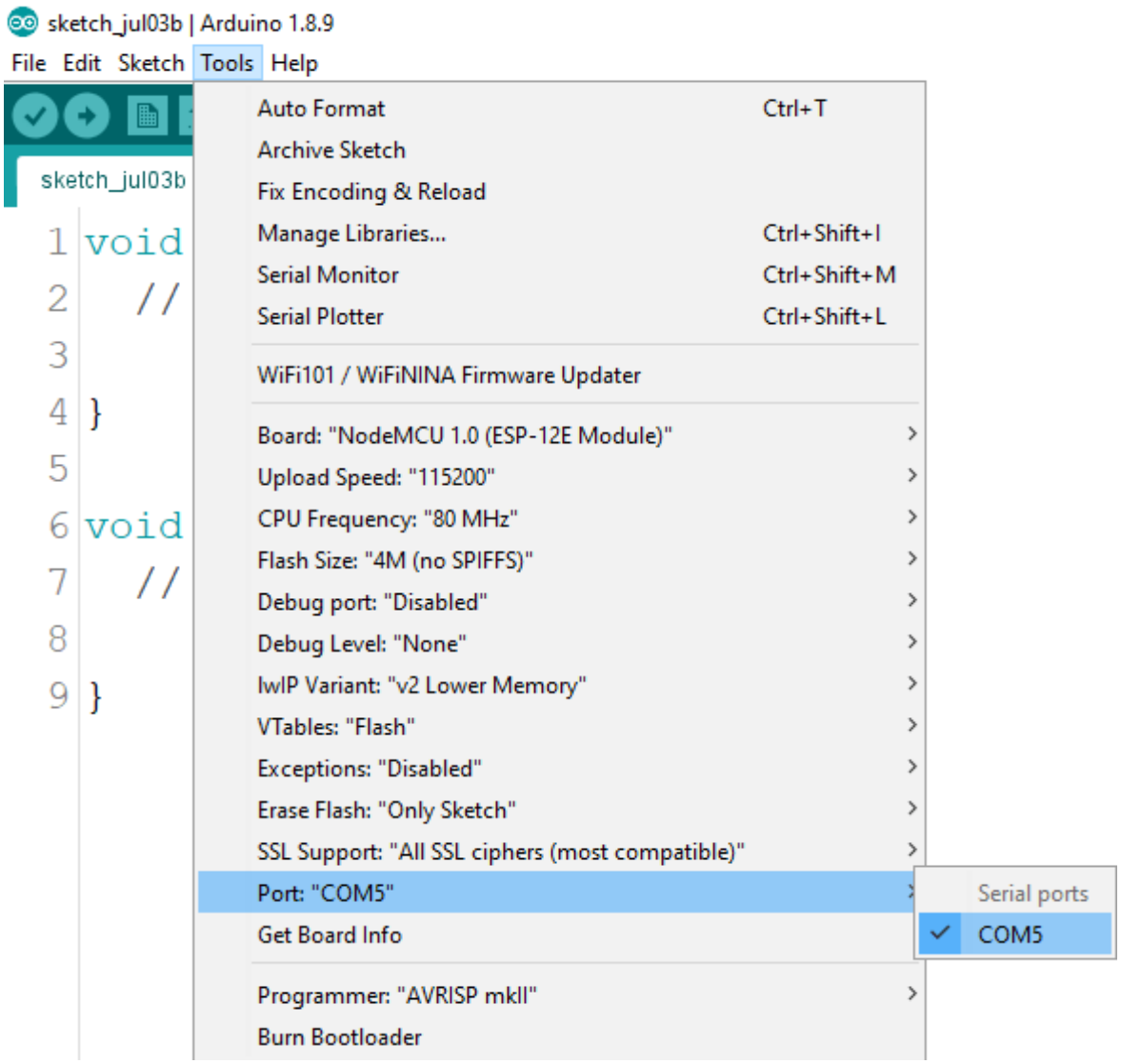
4. Search for **ESP8266** and press install button for the "**ESP8266 by ESP8266 Community**



5. Choose Your Board



6. You also need to select the Port:



TIME-BASED ONE-TIME PASSWORD (TOTP)

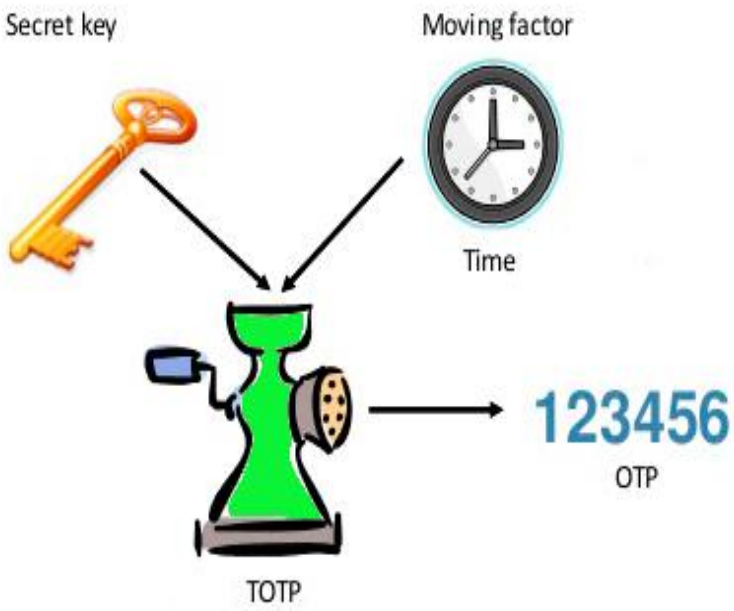
- Time-based One-time Password (TOTP) is a computer algorithm that generates a one-time password (OTP) which uses the current time as a source of uniqueness. An extension of the HMAC-based One-time Password algorithm (HOTP), it has been adopted as Internet Engineering Task Force (IETF) standard RFC 6238.[1]

- TOTP is the cornerstone of Initiative for Open Authentication (OATH), and is used in a number of two-factor authentication (2FA) systems.

TOTP ALGORITHM

- what is TOTP authentication?
  An uncomplicated answer is — it's a 2-factor verification method that uses the time as a variable. Let's expand on this a bit and unravel how TOTP authentication actually operates.

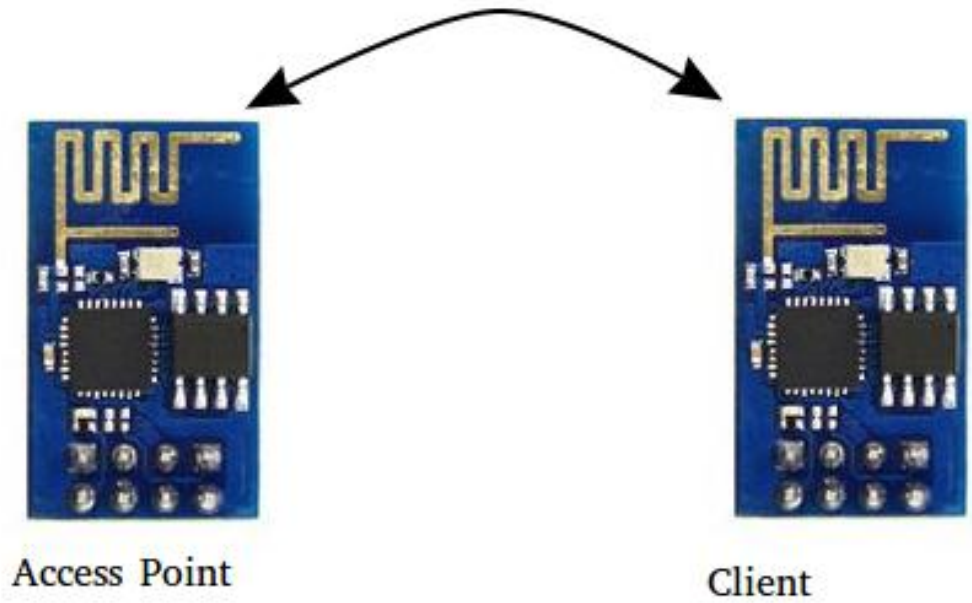
- TOTP algorithm (RFC 6238) implies that an OTP is a product of two parameters encrypted together. These are a common value, which is a shared secret key, or seed; and a variable, in this case – the running time. These parameters are encrypted with a hash function..



Two-Factor Authentication (TFA) protocol

Implementation of the Generic Two-Factor Authentication (TFA) protocol that can be applied to any IoT systems that require enhanced security and authentication eligibility.

today we have implemented a protocol on two NodeMCU devices, one of them is the Client (Station) and the other is the Server(Access point)



2FA CODING

1.SERVER

File Edit Sketch Tools Help

NodeMCU 1.0 (ESP-12E ...

WiFi-Server.ino

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
//#include <ESP8266WiFiMulti.h>
#include <ESP8266mDNS.h>
#include <ESP8266WebServer.h> // Include the WebServer library

#include <Arduino_JSON.h>

char ClientToken[] = "wfsnmyto-n47b-6bhc-7fam-4ltzatyixcmyg5";
String LastApiKey = "";

/* Put your SSID & Password */
const char *ssid = "NodeMCU";      // Enter SSID here
const char *password = "12345678"; //Enter Password here

/* Put IP Address details */
IPAddress local_ip(192, 168, 1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);

ESP8266WebServer server(80);

unsigned long previousKeyCreationTime = 0;
const unsigned int KeyValidityInterval = 120000;

void setup()
{
  Serial.begin(9600);
  Serial.println("\n\n");

  WiFi.softAP(ssid, password);
```

2FA CODING

1.SERVER

File Edit Sketch Tools Help

NodeMCU 1.0 (ESP-12E ...

WiFi-Server.ino

```
  WiFi.softAP(ssid, password);
  WiFi.softAPConfig(local_ip, gateway, subnet);
  delay(100);
  Serial.print("IP address:\t");
  Serial.println(WiFi.softAPIP()); // Send the IP address of the ESP8266 to the computer

  server.on("/data", HTTP_POST, handle_OnData);
  server.on("/login", HTTP_POST, handle_Login);
  server.onNotFound(handle_NotFound);

  server.begin();
  Serial.println("HTTP server started");
  //  generateRandomKey();
}
void loop()
{
  server.handleClient();
  MDNS.update();
}

void handle_Login()
{ // If a POST request is made to URI /login
  Serial.println();
  Serial.println("*******Login Request!*******");

  if (server.hasArg("plain") == false)
  { //Check if body received
    Serial.println("No Body Received");
    JSONVar myObject;
    myObject["Error"] = "No Body Received";
```

2FA CODING

2.CLIENT

File Edit Sketch Tools Help

NodeMCU 1.0 (ESP-12E ...

Wifi-Client.ino

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti WiFiMulti;

#include <Arduino_JSON.h>

// Replace with your network credentials
const char *ssid = "NodeMCU";       // Enter SSID here
const char *password = "12345678"; //Enter Password here

//Your IP address or domain name with URL path
const char *serverLogin = "http://192.168.1.1/login";
const char *serverPostData = "http://192.168.1.1/data";

String temperature;
int ResponseCode = 0;
char MyToken[] = "wfsnmyto-n47b-6bhc-7fam-4ltzatyixcmyg5";
String LastApiKey = "";

unsigned long previousMillis = 0;
const long interval = 20000;

bool isLoggedInFlag = 0;

void setup()
{
  Serial.begin(9600);
  Serial.println();
```

2FA CODING

2.CLIENT

File Edit Sketch Tools Help

NodeMCU 1.0 (ESP-12E ...

Wifi-Client.ino

```
  delay(30);

  WiFi.mode(WIFI_STA);
  WiFiMulti.addAP(ssid, password);
  while ((WiFiMulti.run() == WL_CONNECTED))
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Connected to WiFi");
  isLoggedInFlag = 0;
}

void loop()
{
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval)
  {
    Serial.println();
    Serial.println("*******Inside Loop*******");
    if ((WiFiMulti.run() == WL_CONNECTED))
    {
      if (!isLoggedInFlag)
        doLogin();
      else
        sendData();

      previousMillis = currentMillis;
    }
    else
```

2FA .RESULTS

1.SERVER

IP address: 192.168.1.1
HTTP server started
*******Login Request!*******
Body received:
{"username":"admin","password":"password123","Token":"wfsnmyto-n47b-6bhc-7fam-4ltzatyixcmyg5"}
Initial Token validated | Sending API Key
Here is your random string: yyu0agfi-rygv-3xru-bf30-cbjvhxj3wcwo46
previousKeyCreationTime: 21449
*******Data Receive!*******
Last ApiKey Valid till: 101419
Body received: {"ApiKey":"yyu0agfi-rygv-3xru-bf30-cbjvhxj3wcwo46","data":"25"}
ApiKey validated | Saving your data!
*******Data Receive!*******
Last ApiKey Valid till: 81419
Body received: {"ApiKey":"yyu0agfi-rygv-3xru-bf30-cbjvhxj3wcwo46","data":"21"}
ApiKey validated | Saving your data!

2FA CODING

2.CLIENT

```
Connected to WiFi
*******Inside Loop*******
Trying to login!
HTTP Response code: 202
{"ApiKey":"yyu0agfi-rygv-3xru-bf30-cbjvhxj3wcwo46"}
Accepted
*******Inside Loop*******
Sending Data!
HTTP Response code: 201
{"Status":"Data Received!"}
Created
*******Inside Loop*******
Sending Data!
HTTP Response code: 201
{"Status":"Data Received!"}
Created
```

Next Step

Addition optimization to the system

Thank you