INTERNET OF OBJECTS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

DEPARTMENT OF COMPUTER ENGINEERING AND INFORMATION TECHNOLOGY

FINAL PROJECT

ELECTRIC SIGNAL LIGHT

TEAM / WORK DETAILS

STUDENT DETAILS 1: ATHANASIOU VASILIOS EVANGELOS – 19390005 – PADA
STUDENT DETAILS 2: MANTZOUKAS ANGELOS-VASILIOS – 19390128 – PADA
STUDENT DETAILS 3: PYLARINOS CHRISTOS – 20390278 – PADA

LABORATORY LEADER: GAROFALAKI ZAKHARENIA
THEORY LEADER: KARKAZIS PANAGIOTIS

INTERNET OF OBJECTS

CONTENTS

INTERNET OF OBJECTS

1. Introduction

The work concerns the study of an electric traffic light whose operation is defined by the Arduino UNO microcontroller and the data is stored in real-time on the ThingSpeak platform. The implementation and development of this IoT application requires certain specifications which are:

- **Understanding microcontrollers and their interaction with electronic components** : In short, it requires an understanding of hardware that includes analog and digital inputs/outputs, such as the Arduino UNO microcontroller and the ESP-01 WiFi module.
- **Networking and data exchange** : Connecting the Arduino via WiFi and exchanging data with channels of the ThingSpeak platform, both for visualization and for sending/receiving data.
- **Graphical data visualization** : The use of graphical elements to present data and system status.
- **Data reading and writing:** The programming ability to read and send real-time data.

1.1 Reference to basic concepts

The basic concepts used in the operation of the electric traffic light are as follows:

- **Microcontrollers (Arduino)** : Microcontrollers are small, flexible devices used to control electronic circuits. In this project, the Arduino UNO microcontroller is responsible for switching the light indicators of a traffic light and communicating with the ThingSpeak platform.
- **Internet of Things (IoT)** : IoT refers to the connection of physical objects to the internet, allowing data to be exchanged between them and other systems. Here, traffic light readings are shared and displayed via the ThingSpeak platform.
- **ThingSpeak** : It is an IoT platform that allows the collection, storage, analysis and visualization of data from connected devices in real time. In this work, it is used to send data from Arduino and display the readings.
- **Protocols and APIs** : Communication with ThingSpeak is done through protocols and APIs, which allow for secure data exchange.
- **ESP-01 (WiFi Module)** : A simple and inexpensive module used to connect Arduino to a wireless network, allowing internet access and communication with ThingSpeak.

2. Requirements and Analysis

2.1 Requirement A: Electric traffic light

2.1.1 Description of functions

The traffic light will operate following the classic color sequence (red, green, orange), with the **Arduino** controlling the duration of each indication: red and green for 30 seconds, orange for 20 seconds. This process will be repeated continuously without interruption.

At the same time, the operation of the traffic light will be displayed on the **ThingSpeak platform** via an online connection to the **ESP-01** . Thus, users will be able to monitor the status of the traffic light in real time, remotely, via the internet.

2.1.2 Using Arduino and ThingSpeak platform

For the creation of the electric traffic light, the combined use of the **Arduino microcontroller and the ThingSpeak** platform ensures the monitoring and visualization of the traffic light data in real time via the internet. In more detail each:

Arduino UNO

The **Arduino UNO** is the microcontroller that controls the traffic light indicators. It is programmed using the **Arduino C language** , and will include the following basic functions:

1. **Controlling Visual Elements in ThingSpeak** : The traffic light consists of three different visual elements (red, green, orange) in ThingSpeak, which are alternated based on the program requirements. The microcontroller will activate each visual element for the corresponding time interval
2. **Timer** : The code must include timers that will measure the duration of each phase of the traffic light (red, green, orange). This is achieved by using the delay() command.
3. **WiFi Connectivity** : The Arduino must be connected to a WiFi network via the ESP-01 so that it can send data to the internet and communicate with the ThingSpeak platform.

ThingSpeak

ThingSpeak platform acts as the online space where the traffic light data is sent and stored. It provides capabilities for monitoring and analyzing the values it receives from the Arduino:

1. **Channel** : A channel is created on the platform where changes in the signal status are recorded. The channel fields correspond to variables that indicate whether the visual element indication is active.
2. **Variables** : Each color of the traffic light is associated with a variable in the ThingSpeak channel. When a visual element lights up, the corresponding variable takes the value 1 for red, 2 for orange, 3 for green, while when it takes the value 0, it goes out.
3. **Visualization** : The data sent by the Arduino can be visualized graphically through the platform. This way, the user can monitor in real time which colors are active and for how long.
4. **Data Communication** : Arduino sends data to the platform via HTTP requests

Data Interconnection and Transmission

The connection of the Arduino to the ThingSpeak platform is achieved through the **ESP-01** , which allows the transmission of data from the microcontroller to the internet. Using HTTP requests, data about the current state of the traffic light is automatically sent to ThingSpeak, where it is stored and displayed in real time.

In this way, the operation of the traffic light can be monitored and recorded remotely, thus ensuring continuous information about the operation of the traffic light.

2.1.3 Requirements for programming and connectivity

To implement the electric traffic light system using **Arduino and the ThingSpeak** platform , specific programming assumptions are required. These requirements include preparing the code to control the optical elements, ensuring proper internet connectivity, and sending data in real time.

Programming the Arduino

The basic operation of the traffic light requires the development of a program on **Arduino** , which will:

1. **Manages the Indications** : The code must control the sequence of the traffic light colors (red, green, orange) and determine the duration of each indication, as required (>=30 seconds for red and green, >=20 seconds for orange).
2. **Adjusts Time Delays** : The duration of each phase of the traffic light is controlled using a time delay, which ensures that the visual elements turn on and off at the correct rate.
3. **Sends Data** : The programming includes commands that allow the Arduino to send data to the ThingSpeak platform, updating the variables that represent the state of each visual element.

Connecting to the ThingSpeak Platform

The Arduino's communication with the **ThingSpeak platform** is the key to seeing the status of the traffic light. To achieve this, you need:

1. **WiFi Connection** : The microcontroller needs to connect to the nearest WiFi access point using the ESP-01 WiFi module. This connection allows the Arduino to send data to the internet, which is then displayed on the ThingSpeak channel.
2. **Secure Data Transfer** : Data must be sent in a secure and reliable manner, using HTTP requests .
3. **Channel Programming in ThingSpeak** : The channel in the platform must be properly configured to receive and display the signal state changes. This includes creating fields for the variables that represent the visual elements.

Communication and Connectivity

Successful connection and continuous operation of the system require reliable communication between the Arduino and the platform. To ensure this, you need:

1. **Stable Internet Connection** : The microcontroller must have permanent access to the network for uninterrupted data transmission.
2. **Continuous Data Update** : The code should provide for regular sending of prices to ThingSpeak so that users can see updates in real time.

The above approach ensures that the system will operate smoothly, and connectivity between the Arduino and the platform will remain stable throughout the operation of the traffic light.

2.2 Task B: Sending data to a channel

2.2.1 Description of objectives and functions

The main goal of the request is to develop a functionality that allows sending data to a channel on the ThingSpeak platform. This functionality concerns sending data to a field that receives indicator values for the operation of the traffic light, and includes the following:

Sending data:

- The value **0 is set to the Field 8** variable of our channel, using HTTP requests to communicate with the ThingSpeak API.
- This action aims to enable data to be sent to a channel.

Suggestions for improvement:

- This specific implementation can be adapted to send different data or communicate with another application channel, supporting the development of larger and more complex IoT systems.

The implementation of the above function is done by using the ESP-01 to connect to WiFi and send HTTP requests via Arduino, following the instructions of the ThingSpeak platform API.

2.2.2 Analysis of programming steps

To implement data sending to our channel, the main steps include:

Connecting to WiFi:

Connecting the Arduino to the WiFi network via the ESP-01 is necessary to send data to ThingSpeak.

Channel commands and information:

- Using HTTP requests (**GET**) to send data to **Field 8** of our channel.
- Our channel's API Key is required (which must be added to the corresponding variable in the **myWriteAPI code**).

Operation Verification:

- The code checks the success of the send via responses from the ThingSpeak API.
- Records messages to the serial monitor for debugging.

2.3 Task C: Reading data from a channel

2.3.1 Description of required settings and functions

To read the data from the Field 8 channel which includes indications for the operation of the signal, the following is required:

- **ESP-01 and Wi-Fi settings**
  - The connection to the Wi-Fi network must be configured via the **AT+CWJAP command** , using the appropriate network details (SSID and password)
  - The ESP-01 must be set to client mode (**AT+CWMODE=1**) to communicate with the ThingSpeak server.
- **API and channel configuration**
  - **Read API Key** is required , which allows access to its data.
  - The channel number (Channel ID) for the specific application channel must be specified.
- **HTTP GET function**
  - Communication with the ThingSpeak API is done via HTTP GET requests, where the channel data is requested in JSON format (we also need the Channel ID).
  - **AT+CIPSTART** command is used to establish a TCP connection with the server, and then the **AT+CIPSEND command** to send the request.
- **Data Analysis (JSON)**
  - The response received from the channel must be processed to extract the necessary information (e.g. the channel fields)

2.3.2 Analysis of programming steps

To implement the requested, we followed the specific steps:

- **Connection configuration**
  - To configure the connection, the appropriate AT commands are sent to initiate communication between the Arduino UNO and ThingSpeak.

- **Writing and Sending an HTTP GET Request**
    - To get the channel data in JSON format, we send the appropriate HTTP request, store the length of the message, and send the request.

- **Receive and analyze response**
    - We read the response from the ESP-01 and store it in a variable.
    - We locate and extract the value from JSON using functions like **indexOf()** and **substring()**
- **Data Handling**
    - Depending on the value extracted from the field, we perform the corresponding function, where with a value of 0 the traffic light remains in operation, while with a value of 1, it is turned off by turning on the orange lighting.
- **Verification of the operation of the traffic light**
    - Every 10 minutes, the out-of-service indication for the traffic light is set to be sent to the field8 channel, in order to check that the setting is working.
- **Close Connection**
    - We close the TCP connection after the request is complete.

3. Hardware Circuit Description

The traffic light circuit is based on an Arduino UNO and uses the ESP-01 for internet connectivity, as shown in the image. The circuit combines the physical display of traffic lights with the ability to be remotely monitored via the ThingSpeak platform.

3.1 Connection Analysis

The wiring is based on an Arduino UNO connected to the ESP-01 via a breadboard. Power is supplied to the ESP-01 from the Arduino, while serial communication between the two modules is achieved via ports 6 and 7.

- The ESP-01 connects to the breadboard and then to the Arduino via cables, allowing data exchange.
- The necessary power signals (VCC, GND) are provided to the module by the Arduino board.

3.2 Components Used and Their Role

- **Arduino:** The microcontroller responsible for controlling the visual elements and for interfacing with the ThingSpeak platform.
- **ESP-01:** The WiFi module that connects the Arduino to the network and allows data to be sent to ThingSpeak.
- **Resistors:** Resistors play an important role in the proper functioning and protection of the connection between the Arduino Uno and the ESP-01 WiFi module. The two resistors create a voltage divider for the TX (Transmit) signal from the Arduino to the RX (Receive) pin of the ESP-01. Its role is to reduce the voltage from 5V to 3.3V to protect the ESP-01 from voltages greater than 3.3 V that can cause damage
- **Connecting via cables:** Cables are used to connect the Arduino to the ESP-01, as well as to provide voltage and ground to the circuit.

3.3 Circuit Diagram

The circuit diagram can be described as follows:

- The Arduino UNO is connected to the ESP-01 via cables for power supply and serial communication.
- The wiring is done on a breadboard, where the cables connect the corresponding ports between the two components.
- The Arduino board provides power and control to the ESP-01 for internet connectivity.

The circuit can be expanded by adding LEDs to display the traffic light indications, as described in the previous steps.

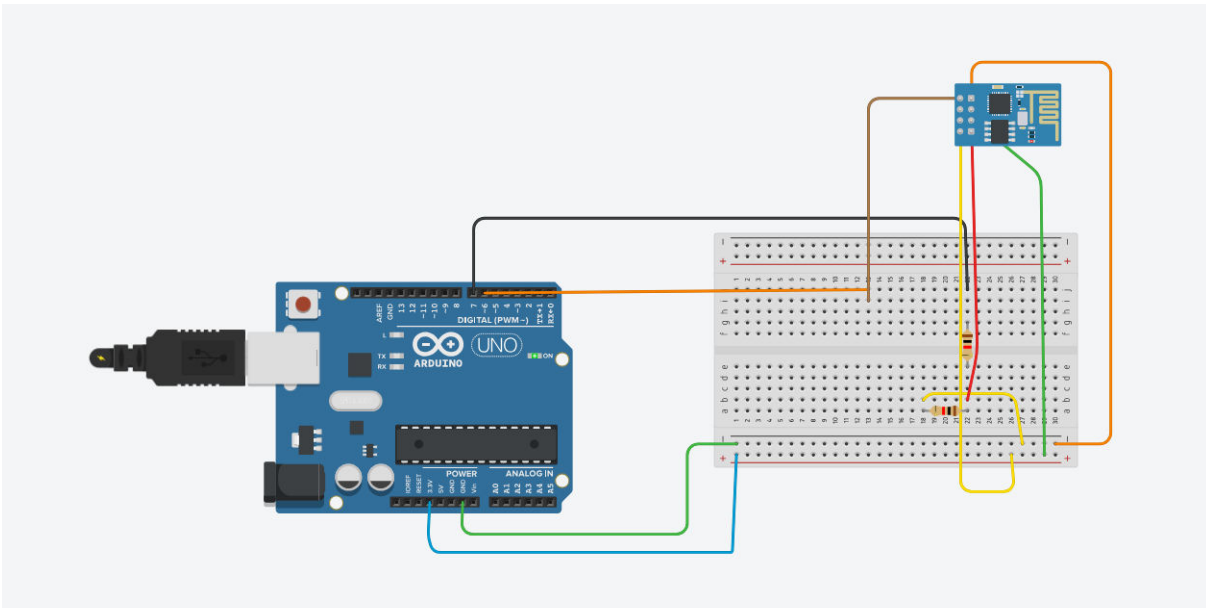
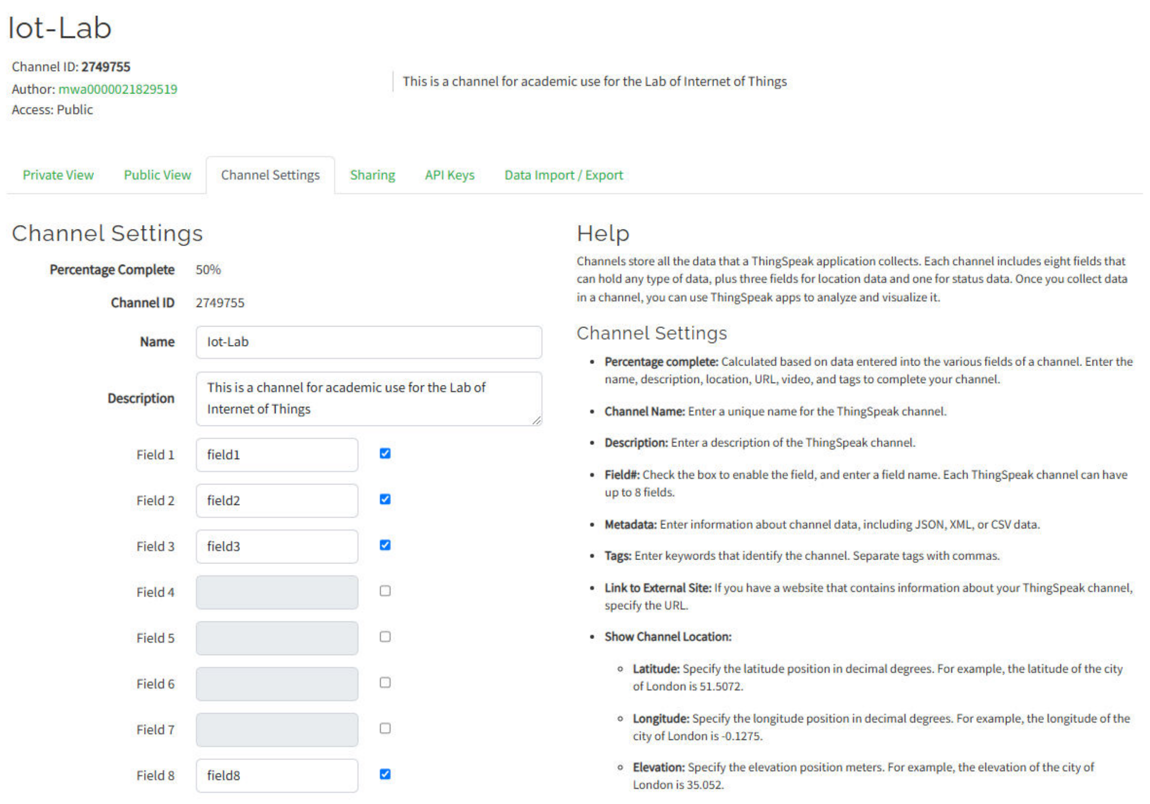


Figure 1. The Arduino UNO circuit with the ESP8266 Wi-Fi module in Tinkercad

4. ThingSpeak Channel Settings

This section describes the ThingSpeak channel settings, as well as the use of visuals and diagrams to illustrate the operation of the traffic light system. The details of the settings and visuals are listed below.

4.1 Explanation of variables and settings



Iot-Lab

Channel ID: **2749755**
Author: mwa0000021829519
Access: Public

This is a channel for academic use for the Lab of Internet of Things

Channel Settings

Percentage Complete	50%	
Channel ID	2749755	
Name	Iot-Lab	
Description	This is a channel for academic use for the Lab of Internet of Things	
Field 1	field1	✓
Field 2	field2	✓
Field 3	field3	✓
Field 4		☐
Field 5		☐
Field 6		☐
Field 7		☐
Field 8	field8	✓

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
  - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.

Figure 2. Variables and settings in Thingspeak.

- **Channel ID** : The channel has the unique number 2749755 and is called Iot-Lab .
- **Author** : The channel was created by the user with the ID number mwa0000021829519 and is available as Public for academic use within the framework of the Internet of Things (IoT) Laboratory.

The channel has eight (8) fields defined, with the first three being used to monitor the traffic light colors and Field 8 being used as the Alert Signal **.**

- **Field 1 (field1)** : Used to indicate the red signal (Red Signal).
- **Field 2 (field2)** : Used to indicate the orange signal .

- **Field 3 (field3)** : Used to indicate the green signal (Green Signal).
- **Field 8 (field8)** : Used for sending and tracking the Alert Signal, which is a monitoring channel for the operation of the traffic light.

The above fields are activated and updated via the ESP-01, which is connected to the Arduino, which in turn controls the traffic light.

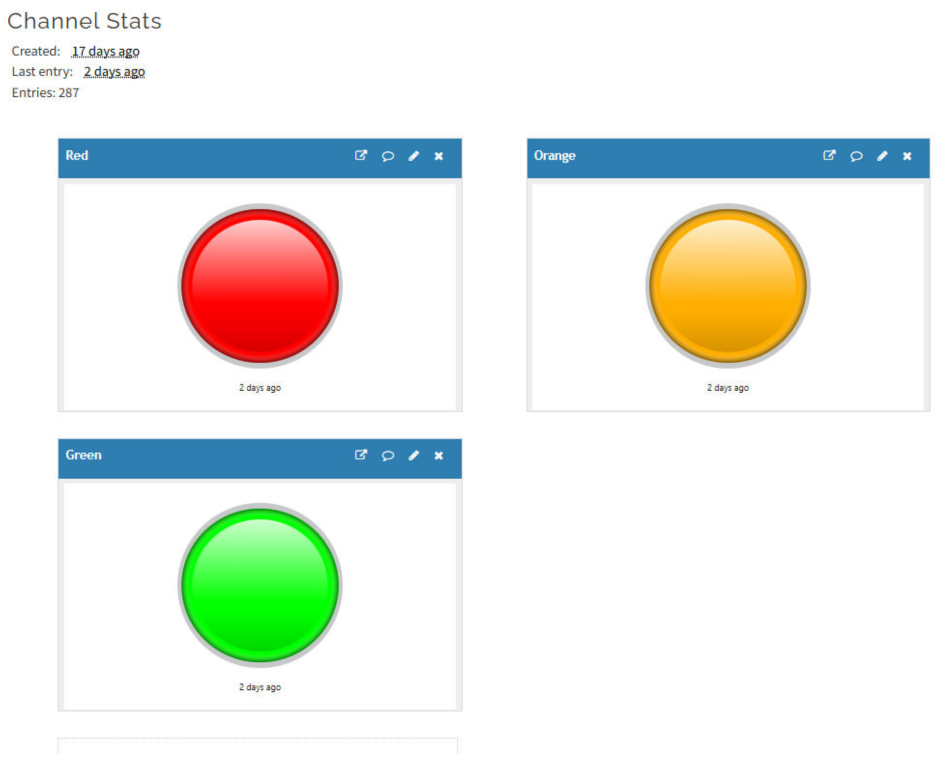
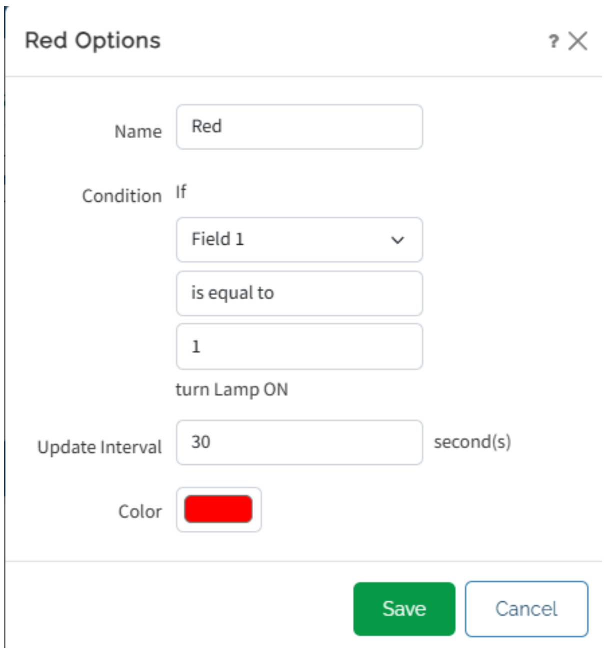


Figure 3. Traffic lights in the fields.

4.2 Description of the use of visual elements

For better understanding and monitoring of data, the channel includes diagrams that capture the status of the traffic light in real time:


Red Options

Name: Red

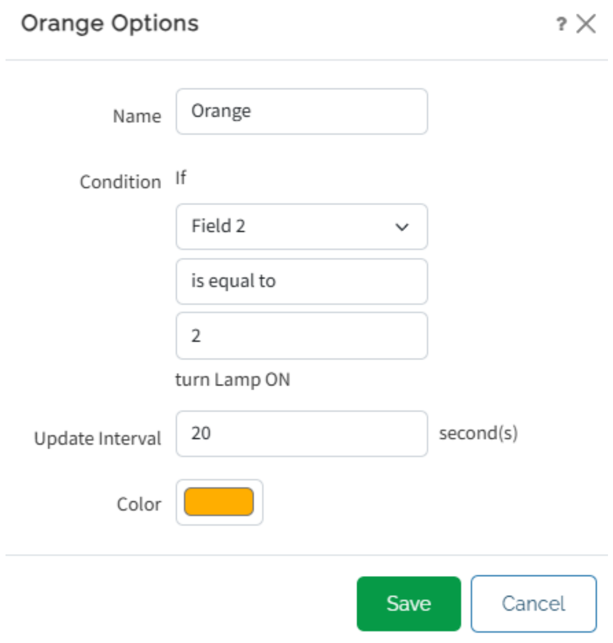
Condition: If Field 1 is equal to 1 turn Lamp ON

Update Interval: 30 second(s)

Color: 🟥

Figure 4. Settings for the red signal.

- **Red Options Widget** :
  - **Name** : Red
  - **Condition** : If Field 1 is equal to 1, then the red light is on.
  - **Update time** : The widget is updated every 30 seconds .
  - **Color** : The color displayed is red for easy identification.


Orange Options

Name: Orange

Condition: If Field 2 is equal to 2 turn Lamp ON

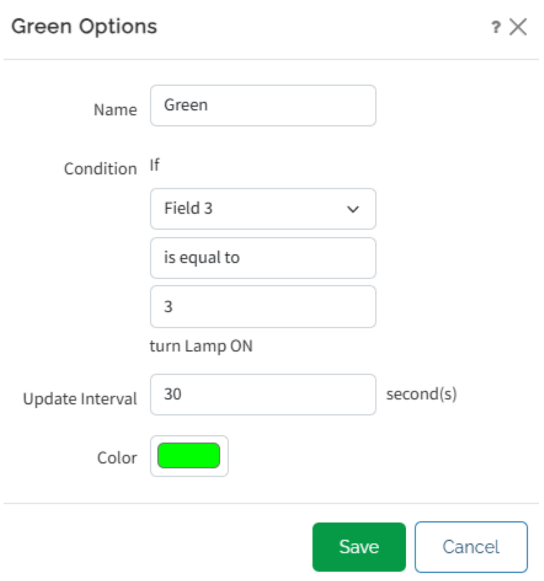
Update Interval: 20 second(s)

Color: 🟧

Figure 5. Settings for the orange signal.

- **Orange Options Widget** :
  - **Name** : Orange

- **Condition** : If Field 2 is equal to 2, then the orange light turns on.
- **Update time** : The widget is updated every 20 seconds .
- **Color** : The color displayed is orange.



Green Options

Name	Green
Condition	If
	Field 3
	is equal to
	3
	turn Lamp ON
Update Interval	30 second(s)
Color	🟩

Figure 6. Settings for the green signal.

- Green Options Widget :
  - **Name** : Green
  - **Condition** : If Field 3 is equal to 3, then the green light is on.
  - **Update time** : The widget is updated every 30 seconds .
  - **Color** : The color displayed is green.

- **Variable charts** :

There are four charts that depict the field data:

- **Field 1 Chart (Red Signal)** : Displays the history of values for the red light, which shows when the light is on (value 1).
- **Field 2 Chart (Orange Signal)** : Displays the price history for the orange light, which shows when the light is on (value 2).
- **Field 3 Chart (Green Signal)** : Displays the history of values for the green light, which shows when the light is on (value 3).
- **Field 8 Chart (Alert Signal)** : Shows the status of the Alert Signal, which is used as a notification by the system (value 0 or 1) for the operation of the traffic light.

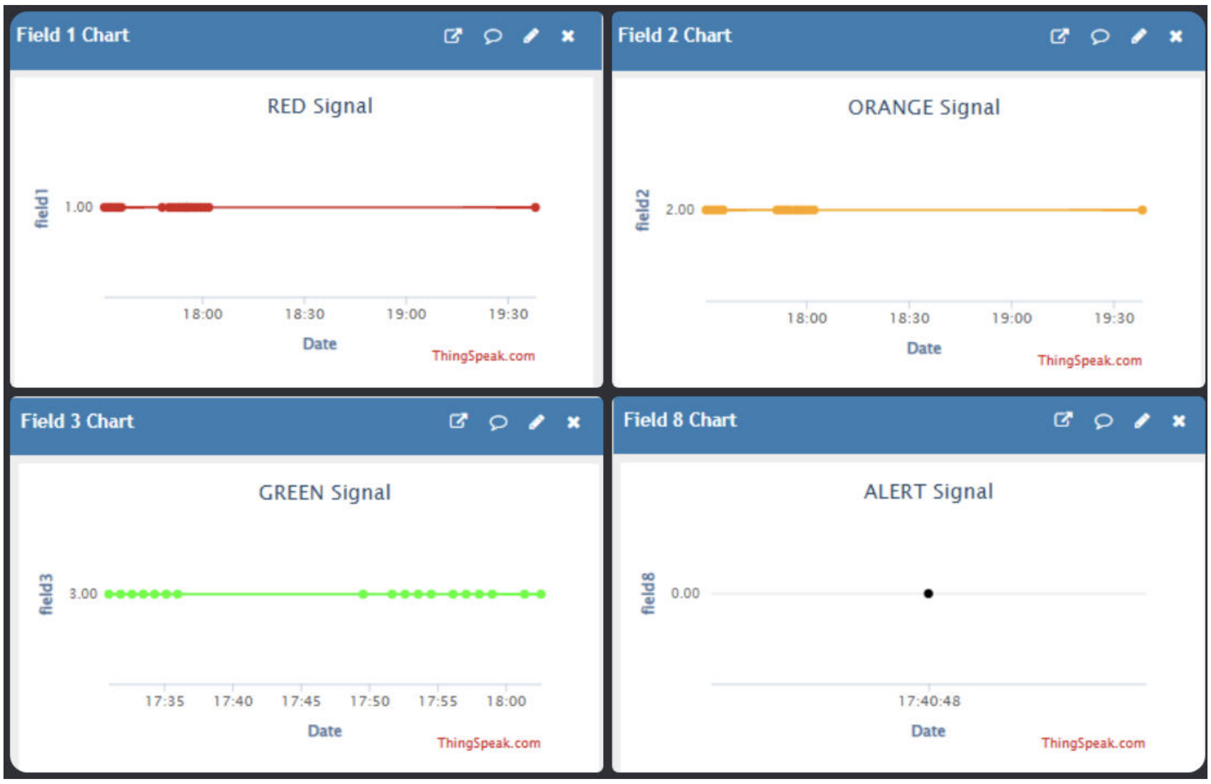


Figure 7. The diagrams that depict the field data.

5. Programming Code

5.1 Analysis of the code used

The code implements a traffic light control system using Arduino UNO that communicates over the internet with ESP-01. The system communicates with the ThingSpeak platform, allowing data to be read and written via channel fields. It includes functions for managing the visual elements of the traffic light, as well as the ability to monitor the operation of the traffic light via a channel field that, depending on the values it reads, disables the traffic light for a certain period of time (alert field).

The main features of the code are as follows:

- **Communication via ESP-01:** Used to send and receive data from ThingSpeak.
- **Traffic light management:** Adjusts traffic light colors (red, green, orange) with defined time delays.
- **Out of order status:** Checks and updates field8 to change the traffic light mode to amber if necessary.
- **Periodic updates:** Every 10 minutes (600,000 ms) the field8 field in ThingSpeak is updated with the value 1 to check the "out of order" status function.

5.2 Commenting and explaining what each piece of code does

Below is an analysis of the key parts of the code:

Configuring initial parameters

```
MySSID string = " linksys_1 " ;
String myPWD = "" ;
String myHOST = " api.thingspeak.com " ;
String myPORT = " 80 " ;
String myWriteAPI = " 6UKE7N1W16R0TIOC " ;
String myReadAPI = " MCX66QIC4S1OR75T " ;
String myCHANNEL = " 2749755 " ;
```

The basic WiFi and ThingSpeak connection parameters are defined.

- **mySSID** ▯ WiFi network name
- **myPWD** ▯ password for the WiFi network
- **myHOST** ▯ host name to which Arduino will connect (ThingSpeak)
- **myPORT** ▯ door that will become the communication channel
- **myWriteAPI** ▯ API key where we gain access to send data
- **myReadAPI** ▯ API key where we gain access to read data
- **myCHANNEL** ▯ Channel ID where information will be exchanged

ESP-01 Settings

```
espData ( " AT+RST " , 1000 , DEBUG );
espData ( " AT+CWMODE=1 " , 1000 , DEBUG );
espData ( " AT+CWJAP= \" " + mySSID + " \" , \" " + myPWD + " \" " , 1000 ,
DEBUG

while (! espSerial . find ( " WIFI GOT IP " ))
{
      Serial.print ( " . " ) ;
      delay ( 1000 );
      Serial.print ( " . " ) ;
      delay ( 1000 );
      Serial.print ( " . " ) ;
      delay ( 1000 );
}

Serial . println ( " Connected! " );
delay ( 1000 );
```

The basic connection settings of the ESP-01 with the Arduino UNO are defined.

- **AT+RST** ▯ ESP-01 reboots to attempt reconnect
- **AT+CWMODE=1** ▯ Set to client mode for connection
- **AT+CWJAP=”mySSID”,”myPWD”** ▯ Connecting to the WiFi network with the name mySSID and password myPWD

The espData() function is responsible for sending AT commands to the ESP-01 via serial communication and reading its response.

```
String espData ( String command , const int timeout , boolean debug )
{
    Serial . print ( " AT Command ==> " );
    Serial.println ( command ) ;

    response = "" ;
    espSerial . println ( command );
    long int time = millis ();
    while (( time + timeout ) > millis ())
    {
        while ( espSerial . available ())
        {
            character c = espSerial.read ( ) ;
            response += c ;
        }
    }
    if ( debug )
    {
// Serial. print(response);
    }

    return response ;
}
```

Visual Asset Management (Required A4)

```
setTrafficLight ( " RED " );
delay ( DELAY_RED );
setFieldValue ( fieldRed , myWriteAPI , 0 );

setTrafficLight ( " GREEN " );
delay ( DELAY_GREEN );
setFieldValue ( fieldGreen , myWriteAPI , 0 );

setTrafficLight ( " ORANGE " );
delay ( DELAY_ORANGE );
setFieldValue ( fieldOrange , myWriteAPI , 0 );
```

The setTrafficLight() function determines which color will be activated:

- **RED** → When we send the value 1 to field1, then the red light turns on
- **ORANGE** → When we send the value 2 to field2, then the orange light turns on
- **GREEN** → When we send the value 3 to field3, the green light turns on.

The indicators light up after specified time delays defined in the corresponding variables. Specifically, 30 sec. for red and green and 20 sec. for orange. By sending the value 0 to the corresponding fields, we turn off the indicator.

The value is sent via the setFieldValue() routine which will be analyzed later.

```
void setTrafficLight ( String color )
{
String field ;
    if ( color == " RED " 
    {
        field = fieldRed ;
        sendVal = 1 ;
    }
    else if ( color == " ORANGE "
    {
        field = fieldOrange ;
        sendVal = 2 ;
    }
    else if ( color == " GREEN "
    {
        field = fieldGreen ;
        sendVal = 3 ;
    }
    else return ;

    setFieldValue ( field , myWriteAPI , sendVal );
    Serial . println ( " Traffic Light is set to " + color );
}
```

Sending data to ThingSpeak (Request B)

```
setFieldValue ( fieldAlert , myWriteAPI , 0 );
Serial . println ( " ALERT Field set to 0. " );
```

The setFieldValue() function uses the appropriate HTTP GET request to send data to the field8 field that is responsible for monitoring the operation of the signal. It connects to the ThingSpeak API via TCP and sends the value 0 to the field8 field that signals the operating status of the signal. With the help of the espData() routine, the appropriate AT commands for the transmission are sent. Specifically, the following are sent:

- **AT+CIPMUX=1** → Enable multi-connection mode on ESP-01

- **AT+CIPSTART=0, “TCP”, “myHOST”, “myPORT”** ▯ Initiate TCP connection to the remote server (in our case ThingSpeak)
- **AT+CIPSEND=0, “request.length”** ▯ Preparing the ESP-01 to send data over an open TCP connection
- **AT+CIPCLOSE=0** ▯ Terminating the TCP connection

```
void setFieldValue ( String field , String writeAPI , int value )
{
    sendData = " GET /update?api_key= " + writeAPI + " & " + field + " = " +
String ( value );
    espData ( " AT+CIPMUX=1 " , 1000 , DEBUG );
    espData ( " AT+CIPSTART=0, \" TCP \" , \" " + myHOST + " \" , " + myPORT ,
1000 , DEBUG );
    espData ( " AT+CIPSEND=0, " + String ( sendData . length () + 4 ), 1000 ,
DEBUG );
    espSerial . find ( " > " );
    espSerial . println ( sendData );
    Serial . println ( " Value to be sent: " );
    Serial.println ( value ) ;

    espData ( " AT+CIPCLOSE=0 " , 1000 , true );
    delay ( 10000 );
}
```

Reading data from ThingSpeak (Required C1)

```
x01 = getFieldValue ( fieldAlert );
Serial . println ( " ALERT field value: " + x01 );

if ( x01 == 1 )
{
      Serial . println ( " ALERT: Traffic Light out of order. " );
      setTrafficLight ( " ORANGE " );
}
```

The getFieldValue() function uses the appropriate HTTP GET request to read data from a ThingSpeak field. It looks up the position of the field in the JSON response and returns the corresponding value. The field it reads is field8, which is the field that controls the operation of the traffic light. If the value it reads is 1, then it sets the traffic light out of operation with the setTrafficLight() routine by turning on the orange indicator. With the help of the espData() routine, the appropriate AT commands for reading are sent. Specifically, the following are sent:

- **AT+CIPMUX=1** ▯ Enable multi-connection mode on ESP-01
- **AT+CIPSTART=0, “TCP”, “myHOST”, “myPORT”** ▯ Initiate TCP connection to the remote server (in our case ThingSpeak)
- **AT+CIPSTO=10** ▯ Sets the timeout for idle TCP connections.

- **AT+CIPSEND=0, “request.length”** ▯ Preparing the ESP-01 to send data over an open TCP connection
- **AT+CIPCLOSE=0** ▯ Terminating the TCP connection

Periodic update of the value of field8 (Required C2)

```
currentMillis = millis ();
if ( currentMillis - previousMillis >= timer )
{
      previousMillis = currentMillis ;
      Serial . println ( " Setting ALERT field value to 1. " );
      setFieldValue ( fieldAlert , myWriteAPI , 1 );
}
```

Every 10 minutes (600,000 ms) the value of field8 is updated with 1 and lasts for 1 minute. This is done to check if the value of field8 is read correctly and the traffic light is turned off by turning on the orange indicator. The value is sent with the help of the setFieldValue() routine.

5.3 Presentation of results

For as long as we had the equipment in the laboratory, we obtained the following results:

- The normal operation of the traffic light, where it alternates between red, green and amber for set intervals.
- Operating mode (Alert) where the field is automatically updated every 10 minutes and if the value 1 is read, then the traffic light is turned off.
- The AT commands and responses of the ESP-01 for communicating with the Arduino UNO.

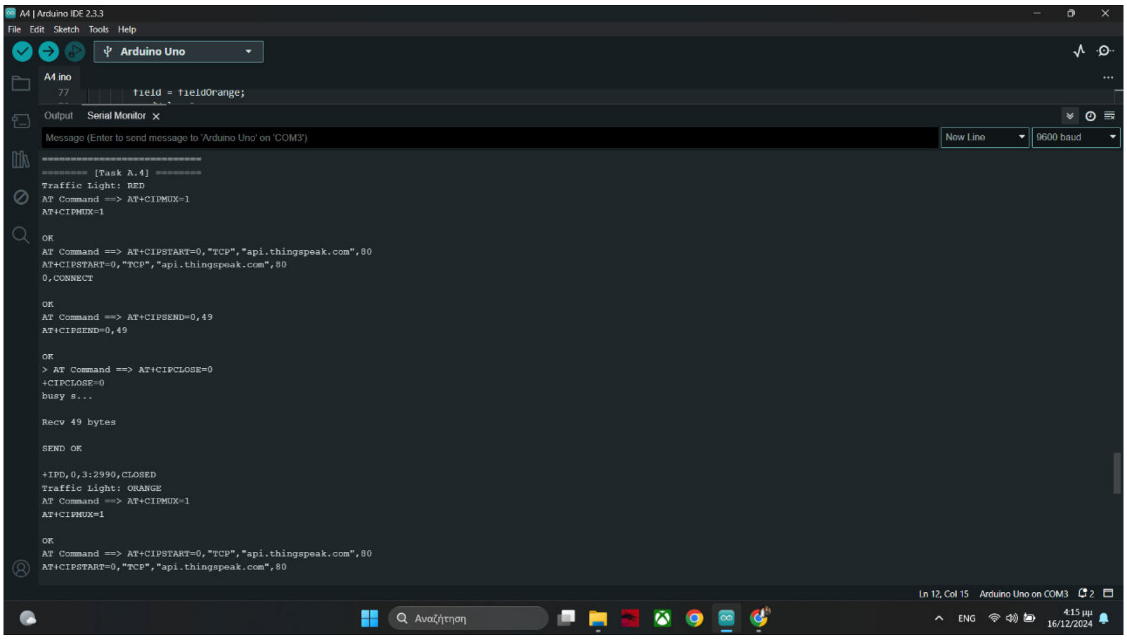

```
============================
======== [Task A.4] ========
Traffic Light: RED
AT Command ==> AT+CIPMUX=1
AT+CIPMUX=1

OK
AT Command ==> AT+CIPSTART=0,"TCP","api.thingspeak.com",80
AT+CIPSTART=0,"TCP","api.thingspeak.com",80
0,CONNECT

OK
AT Command ==> AT+CIPSEND=0,49
AT+CIPSEND=0,49

OK
> AT Command ==> AT+CIPCLOSE=0
+CIPCLOSE=0
busy s...

Recv 49 bytes

SEND OK

+IPD,0,3:2990,CLOSED
Traffic Light: ORANGE
AT Command ==> AT+CIPMUX=1
AT+CIPMUX=1

OK
AT Command ==> AT+CIPSTART=0,"TCP","api.thingspeak.com",80
AT+CIPSTART=0,"TCP","api.thingspeak.com",80
```

Figure 8. Request A4

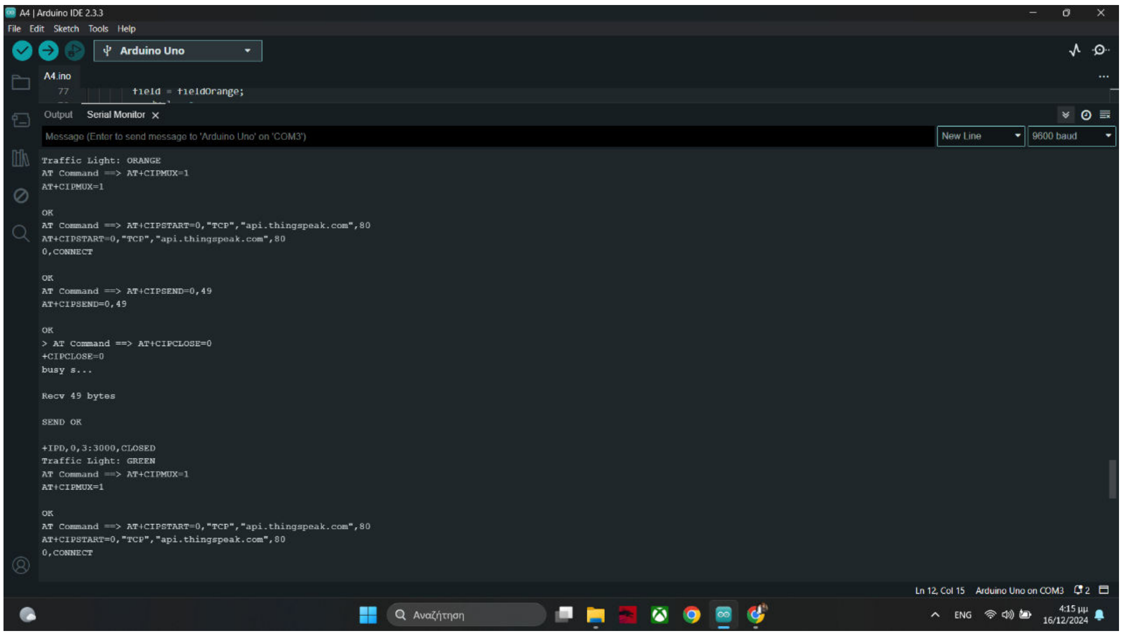
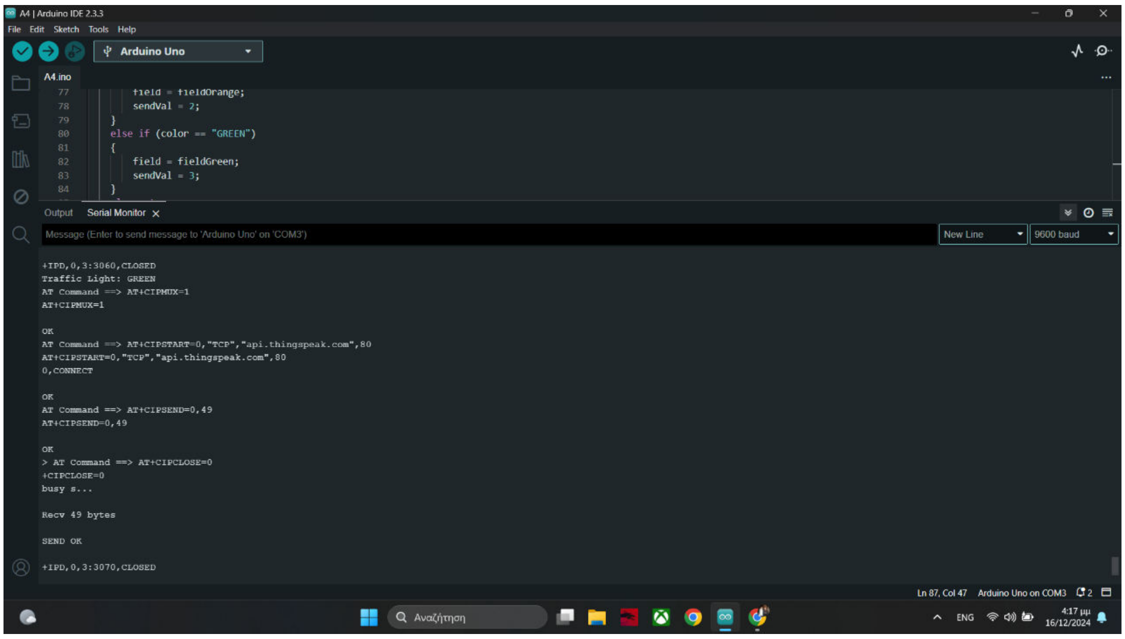

```
Traffic Light: ORANGE
AT Command ==> AT+CIPMUX=1
AT+CIPMUX=1

OK
AT Command ==> AT+CIPSTART=0,"TCP","api.thingspeak.com",80
AT+CIPSTART=0,"TCP","api.thingspeak.com",80
0,CONNECT

OK
AT Command ==> AT+CIPSEND=0,49
AT+CIPSEND=0,49

OK
> AT Command ==> AT+CIPCLOSE=0
+CIPCLOSE=0
busy s...

Recv 49 bytes

SEND OK

+IPD,0,3:3000,CLOSED
Traffic Light: GREEN
AT Command ==> AT+CIPMUX=1
AT+CIPMUX=1

OK
AT Command ==> AT+CIPSTART=0,"TCP","api.thingspeak.com",80
AT+CIPSTART=0,"TCP","api.thingspeak.com",80
0,CONNECT
```

Figure 9. Required A4 (continued)


A4 | Arduino IDE 2.3.3
File Edit Sketch Tools Help
Arduino Uno
A4.ino
Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM3')
New Line
9600 baud
Ln 87, Col 47 Arduino Uno on COM3
Αναζήτηση
4:17 μμ
16/12/2024

```
            field = fieldOrange;
            sendVal = 2;
        }
        else if (color == "GREEN")
        {
            field = fieldGreen;
            sendVal = 3;
        }
```

```
+IPD,0,3:3060,CLOSED
Traffic Light: GREEN
AT Command ==> AT+CIPMUX=1
AT+CIPMUX=1

OK
AT Command ==> AT+CIPSTART=0,"TCP","api.thingspeak.com",80
AT+CIPSTART=0,"TCP","api.thingspeak.com",80
0,CONNECT

OK
AT Command ==> AT+CIPSEND=0,49
AT+CIPSEND=0,49

OK
> AT Command ==> AT+CIPCLOSE=0
+CIPCLOSE=0
busy s...

Recv 49 bytes

SEND OK

+IPD,0,3:3070,CLOSED
```

Figure 10. Required A4 (continued)

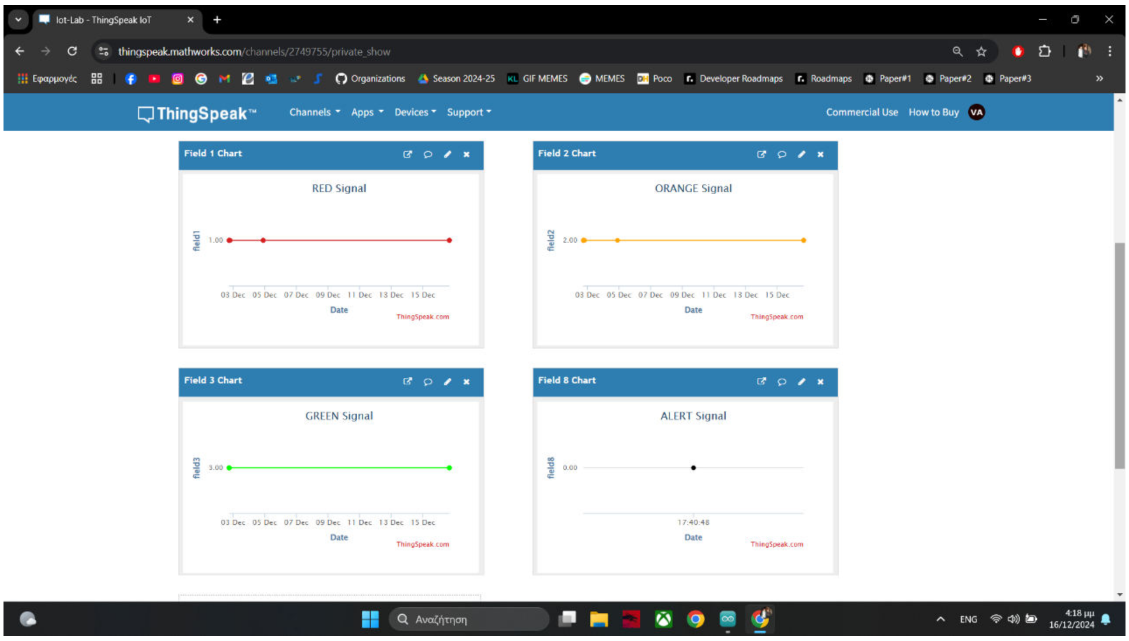


Figure 11. Data in the fields of visual elements in ThingSpeak

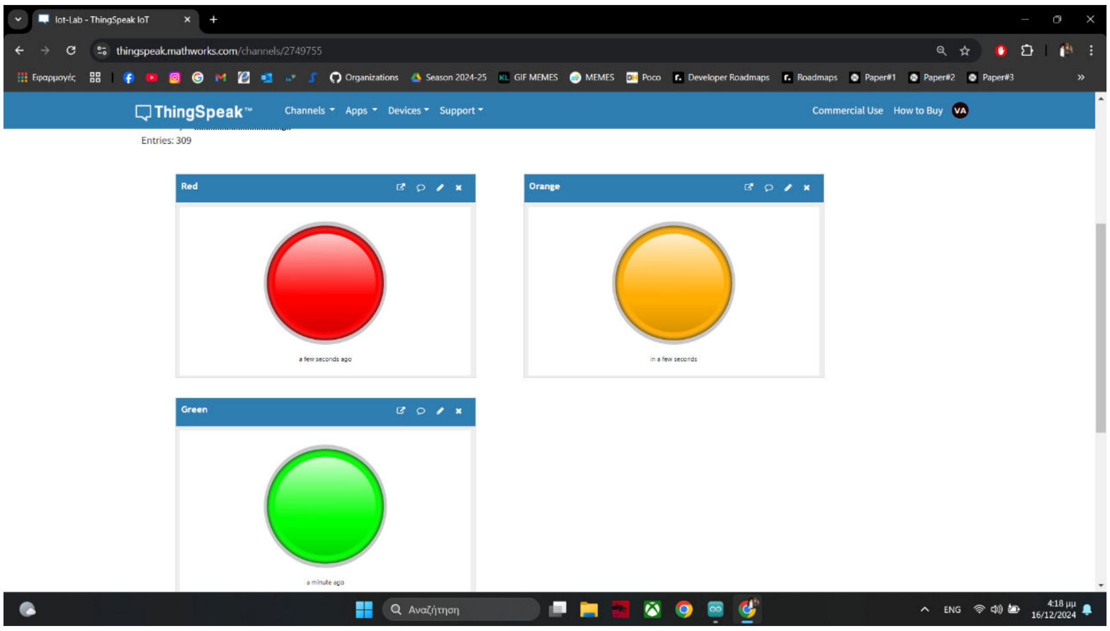
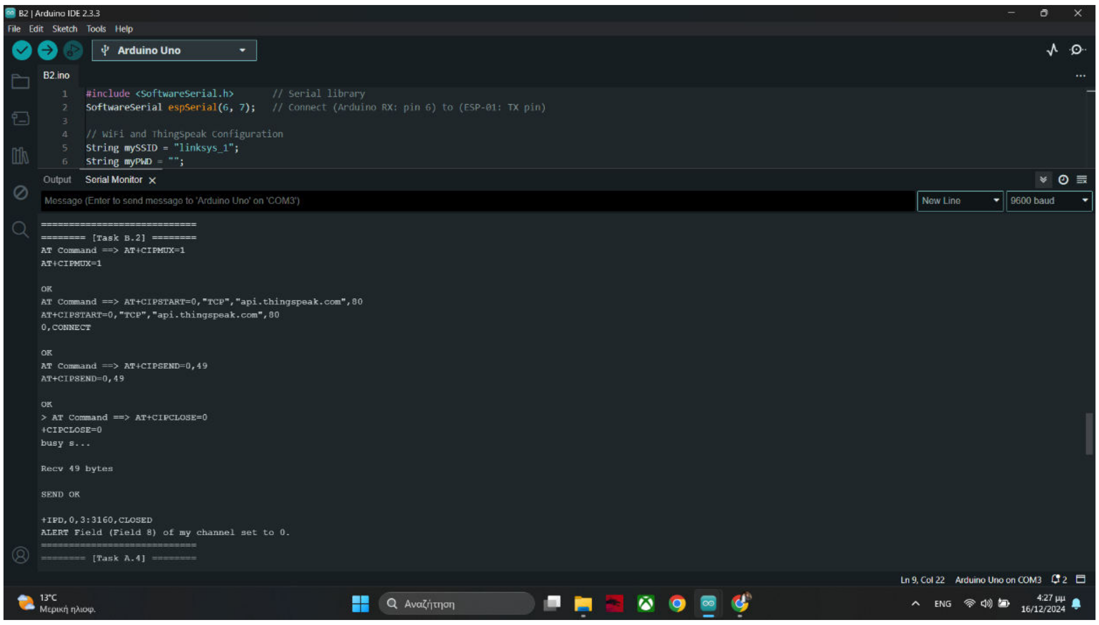


Figure 12. The visual elements in ThingSpeak


B2 | Arduino IDE 2.3.3
File Edit Sketch Tools Help
Arduino Uno
B2.ino
Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM3')
New Line
9600 baud
Ln 9, Col 22 Arduino Uno on COM3

```
#include <SoftwareSerial.h>       // Serial library
SoftwareSerial espSerial(6, 7);   // Connect (Arduino RX: pin 6) to (ESP-01: TX pin)

// WiFi and ThingSpeak Configuration
String mySSID = "linksys_1";
String myPWD = "";
```

```
============================
======== [Task B.2] ========
AT Command ==> AT+CIPMUX=1
AT+CIPMUX=1

OK
AT Command ==> AT+CIPSTART=0,"TCP","api.thingspeak.com",80
AT+CIPSTART=0,"TCP","api.thingspeak.com",80
0,CONNECT

OK
AT Command ==> AT+CIPSEND=0,49
AT+CIPSEND=0,49

OK
> AT Command ==> AT+CIPCLOSE=0
+CIPCLOSE=0
busy s...

Recv 49 bytes

SEND OK

+IPD,0,3:3160,CLOSED
ALERT Field (Field 8) of my channel set to 0.
============================
======== [Task A.4] ========
```

Figure 13. Desired B

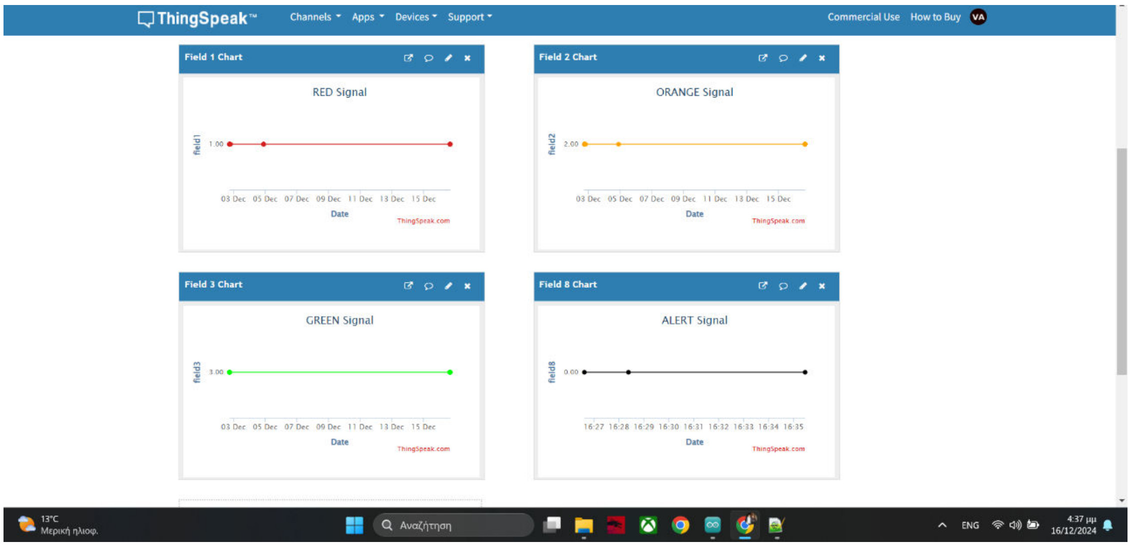


Figure 14. The alert field data in ThingSpeak

5.4 Problems identified and solutions tried

The main problem we encountered was in request C1 where we were reading the value of field8 which monitors the operating status of the signal. While, from request B we were sending the default value 0 which signals the valid operating status, the JSON response was not returning any value.

We tried increasing the delays between the HTTP GET requests we made in the previous queries, in case there was an issue with the value update, but it didn't change the result. The value 0 as seen in the results was normally present in the field8 data. We also changed the condition

```
if ( x01 == 1 )
```

in

```
if ( x01 . equals ( " 1 " ))
```

Maybe it was because the value was stored as a String. In the end, since the last lab we were unable to fix this problem, so we left it as it is, due to lack of equipment.

6. Conclusions

6.1 Review of the work

The work is an important step towards the development of an IoT system and real-time data monitoring. It achieved all the goals we mentioned above and highlighted the potential of the Internet of Things. We gained the appropriate experience both from the development and through the failures, where we pondered and tested solutions.

6.2 Suggestions for improvements

The improvements we would suggest would be to limit HTTP requests, so that there is a period of time between field values being updated and we can have correct results for the operation of the traffic light. Regarding other scalability issues, we could use HTTPS requests for encryption, a reconnect routine in cases of WiFi connection interruption and various others that are not within the scope of the work, but it would be good to implement them to improve the efficiency of our system.

```
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier
```

Thank you for your attention.