# IOT-SUM24

# Faculty of Computers and Data Science

## 02-24-03205 - Field Training

### Module 1 Assignment Report

Hazem Ahmed Abdelfatah – 2203175

Ahmed Mohamed Gaber – 2203173

Youssef Salah Mostafa – 22010442

Moataz Ali Ramadan – 22033177

Momen Wagdy Hamed- 2206134

# Table of Contents: -


1 – Advantages & Disadvantages of IoT ..... 3

1.1 - Advantages of IoT ..... 3

1.2 - Disadvantages of IoT ..... 3

2 – What is a Gateway ..... 3

2.1 - Definition ..... 3

2.2 - Role in IoT layers ..... 3

3 – Fog computing vs Cloud computing ..... 4

3.1 - Fog computing ..... 4

3.2 - Cloud Computing ..... 4

3.3 - Which is better for IoT ..... 4

4 – Difference between HTTP & MQTT ..... 5

4.1 - HTTP ..... 5

4.2 - MQTT ..... 5

5 – Main types of Programming errors ..... 6

5.1 - Runtime errors ..... 6

5.2 - Logic errors ..... 6

5.3 - Compile-time errors ..... 6

6 – Source code & Object code ..... 6

6.1 - Source code ..... 6

6.2 - Object code ..... 6

7 – Enum data type ..... 6

7.1 - Definition ..... 6

7.2 - Use cases ..... 6

8 – Coding Problems ..... 7

8.1 - Distance Travelled ..... 7

8.2 - Printing C ..... 7

8.3 - Velocity of sound ..... 8

8.4 - Interest Rate..... 9
8.5 - Integer Operations..... 10
8.6 - Miles per Gallon..... 11
8.7 - Gravitational Attraction..... 12
8.8 - Bonus Problem 1..... 13
8.9 - Bonus Problem 2..... 14
References..... 16

# 1 – Advantages & Disadvantages of IoT:

**1.1 - Advantages of IoT:** IoT or Internet of Things has enabled many ideas and solutions that were thought to be impossible, thus it has many benefits and merits, some of which being:

- Enhanced efficiency: IoT devices enable real time monitoring, they allow for enhancing overall efficiency, and reduces user and/or administrator intervention. [1]
- Data collection and analysis; IoT devices generate a load of data, which is incredibly useful for tracking and predicting user behavior and preferences to offer a more suitable and friendly experience to the user. [1]
- Process improvement, allowing companies to reduce checking time and find places on premises in need of servicing without wasting time and money. [2]

**1.2 - Disadvantages of IoT:** Like any other technology on earth, IoT has its own disadvantages to face its own advantages, to decide whether IoT is worth implementing and adopting in a company's work force; a company must weigh the pros and cons, and see what can be done to face detrimental demerits in IoT, these disadvantages include:

- Security concerns: most IoT devices function as a network of devices, which means if a single device is compromised the whole network is susceptible to data leaks and unauthorized access. [1]
- Privacy concerns: IoT devices are good at tracking user preferences and habits, which raises questions about how ethical it is to keep track of and utilize such data. [1]
- High initial costs: implementation of IoT infrastructure in a workplace is often expensive which troubles a lot of small businesses. [1]
- Operational complexity: IoT systems function as a complex network, so in case of a wrong result, the wrong result will be propagated through the entire network, which makes IoT systems difficult to debug. [3]

# 2 – What is a Gateway:

**2.1 - Definition:** A gateway in an IoT system has many different definitions, one of which, and the broadest one, is a physical device or a software program that works to connect various parts of the IoT system, cloud devices and sensors, together [4]. It functions like a simple WIFI router, where it routes the data from the IoT devices to the cloud; it, however, is more complex than that, and IoT gateways tend to be complex. [4].

**2.2 - Role in IoT layers:** IoT infrastructure is divided into 4 layers, each layer plays a significant role in providing a secure and coherent IoT system, these layers include:

- Sensor layer: At this layer, data is read from the sensors and collected for future processing. IoT devices operate at this layer. [5]
- Network / Data accusation layer: At this layer, data is collected from multiple sources or IoT devices and transmitted securely to processing devices, IoT gateways operate here to provide a secure and reliable connection between the processing system and the IoT system. [5]
- Data pre-processing layer: At this layer, the collected data is pre-processed and reduced to provide analytics and insights on the data before it is passed onto cloud processing systems. [5]
- Application layer: At this layer, the data is processed deeply, and in-depth analytics and insights are provided to the user from cloud infrastructure.[5]

## 3 – Fog Computing vs Cloud Computing:

**3.1 - Fog computing:** it is a decentralized computing infrastructure where storage and processing components are placed on the edge of the cloud where end users and sensors reside. It involves using devices with low processing capabilities to share the load on cloud systems. Its main goal is using the cloud only for in-depth and long-term processing.[6] *See Figure 1.*

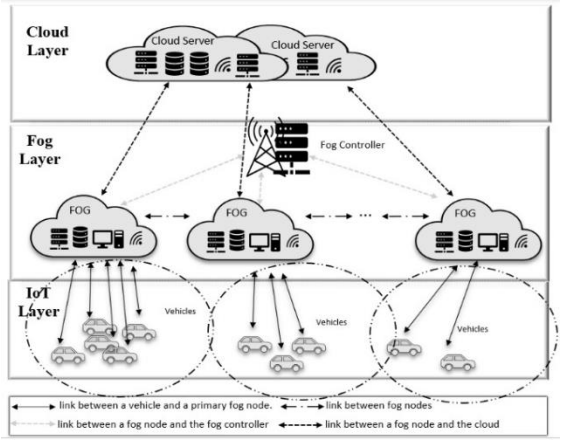


**Figure 1.** Fog computing model [7]

**3.2 - Cloud computing:** it is the use of hosted services such as computing, data storage, servers, and networking, these services are hosted on physical servers which are developed and maintained by the cloud service provider, cloud computing varies in terms of provided service, and how the model is deployed. Some cloud services are hosted on private servers, while others are hosted on public shared servers. Some are kept on premises of the users' company, but the server is maintained by the cloud service provider. [8]

**3.3 - Which is better for IoT:** In case of IoT device, Fog computing is better, since it saves the heavy computing for the cloud, yet it utilizes the edge devices of the cloud, with lower computing power, to perform brief time analytics, and deal with small

problems and faults. These edge devices act as a window to the outside world for the cloud system. Fog computing also has a special subset that was developed with IoT devices in mind, called Edge Computing, where the data is processed in the edge devices as it is generated, such as sensors, cameras, routers, and controllers [6].

## 4 – Difference between HTTP & MQTT:

**4.1 - HTTP:** It is a protocol used for transferring data and documents over the web, short for Hyper Text Transfer Protocol. A client and an end point would communicate with each other through a set of methods, regardless of the method; each interaction functions similarly, where the client sends a request, and the end point returns a response.[9] HTTP methods include several methods such as:

- GET; in this type of method the user tries to access a resource directly from the end point.[9]
- POST; in this type of method the user provides extra information in the request, the end point processes the extra information and returns an appropriate response accordingly.[9]
- HEAD; in this type of method the user requests the headers that would be normally returned if you were to use GET on the given path.[9]
- OPTIONS; in this type of method the user requests the permitted communication methods for a given path. [9]

In IoT devices, using HTTP provides extra overhead that can be removed, there are many reasons why HTTP is not preferred in IoT such as:

- It sends extra information in the packet that is not useful for either party in the communication, the packet size exceeds 8 bytes. [10]
- It requires a TCP connection to send the data over, which takes time to establish so it is not suitable for frequently communicating devices. [10]
- It relies on text encoded data, so encoding and decoding the data repeatedly is causing extra strain on the CPU. [10]
- Its operation model is based on a client-server model which is a one-to-one model, where in most IoT systems a one-to-many model is used. [10]

**4.2 - MQTT**: It is a protocol that was originally designed to enable minimal battery loss and bandwidth usage when connecting to oil pipelines through satellites, it later became a standard in IoT communication for its lightweight communication and quality of data delivery. Its model is based on topics and subscriptions, where a topic is a string and can be nested and contain another topic within. A subscription is given by a client to show which topics concern this device. The broker keeps track of the topic updates and notifies clients that are subscribed to the updated topic. A client can subscribe to multiple topics and a topic can have many subscribers. MQTT also includes multiple Quality of Service levels depending on the urgency of the topic. [10] It also has a message

size of 2 bytes and does not need a pre-existing TCP connection such as with HTTP. [11] When dealing with IoT devices MQTT has become the standard for inter-device communication within the IoT network.

## 5 – Main types of Programming errors:

**5.1 - Runtime errors:** They are errors that occur during the execution of the program they may lead to an incorrect result in a computation, or even lead to the termination of the program, an example is a program that takes user input for division, if the divisor were to equal zero, a runtime error would occur. [12]

**5.2 - Logic errors:** They are errors that do not hinder the execution runtime of the program, but they provide unexpected and incorrect results. They can be difficult to find and deal with since they often appear in unique cases, an example would be an infinite loop with a break condition that is not reached. [12]

**5.3 - Compile-time errors:** They are error that occur during the compilation of the program, they are also known as syntax errors, they are caused by incorrect syntax in the code, they are less common since most modern IDEs notify the user when a syntax error occurs. [12]

## 6 – Source code & Object code:

**6.1 - Source code:** It is written by a developer in a specific language, text-based and human readable, allowing for version control, sharing, error correction, and optimizations. It also allows for sharing across enterprises and collaboration between different developers. [13]

**6.2 - Object code:** It is written by a compiler; it is generated by compiling source code into a language that machines can understand, it is the distributed software that will be reaching the users, allowing for confidentiality since it cannot be read by humans, keeping the source code secret from the users. [13]

## 7 – Enum data type:

**7-1 - Definition:** It stands for enumeration, it is a data type that allows for variables to be defined as predefined constants, where the variable can equal any of the predefined constants. [14] They are useful since they are understood by the programmer defining it and other programmers who come across it.

**7.2 - Use cases:** It is used when dealing with systems that involve predefined constants, such as HIGH, MEDUIM, and LOW. Or NORTH, EAST, SOUTH, and WEST. They

reduce uncertainty in a system since it can only be one of the defined values in a system. [14]

## 8 – Coding Problems:

**8.1 - Travelled Distance:** In the given question, a calculation for the travelled distance in a free drop was required given the time the object has been falling for, the given formula was:

$$dist = \frac{acceleration \cdot time^2}{2}$$

The code for the following problem can be found in *Figure 2.*

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    float time, product, distance, acceleration = 32.0;
    // Tell user to enter the time
    printf("Enter the time in seconds: ");
    scanf("%f", &time);
    // Calculate
    product = (acceleration * time * time);
    distance = product / 2;
    // Output
    printf("The object will drop %.2f feet in %.2f seconds.\n",
distance, time);
    return 0;
}
```

**Figure 2.** C code for Problem 1.

In this code snippet, the user is prompted to enter the time spent in free fall, then the product of the acceleration and the time squared is computed and the distance is calculated in accordance with the given formula.

**8.2 - Printing C:** In the following question, printing the letter C was required, where the user would input a character, and the program prints the letter C out of that character. The code for the problem can be found in *Figure 3.*

In the following code; the user is prompted to enter a character, then the program loops over the vertical grid size and the horizontal grid size. Three main conditions are written for printing the object, the first condition is creating the upper borders for the letter C, the second condition is creating the diagonal curves of the letter, the third condition is creating the sides of the letter.

```
#include <stdio.h>

void main(){
    char obj;
```

```
    int horizontal_grid_size = 5;
    int vertical_grid_size = 10;
    printf("Enter char to print: ");
    scanf("%c",&obj);
    for (int i = 0; i< vertical_grid_size+1;i++){
        for (int j=0; j<horizontal_grid_size+1;j++){
            if ((i == 0 || i == vertical_grid_size) && (j > 1
&& j < horizontal_grid_size)){
                printf("%c",obj);
            }
            else if (i == 1 || i == vertical_grid_size - 1){
                if (j == 1 || j == horizontal_grid_size-1)
printf("%c",obj);
                else printf(" ");
            }
            else if ((i > 0 && i < vertical_grid_size) && (j
== 0 || j == horizontal_grid_size)){
                if (j != horizontal_grid_size)
printf("%c",obj);
                else if (i <= 1 || i == vertical_grid_size-1)
printf("%c",obj);

            }
            else
                printf(" ");
        }
        printf("\n");
    }
}
```

**Figure 3.** C code for Problem 2.

**8.3 - Velocity of sound:** In the following question, the calculation for the velocity of sound was required, given that the velocity of sound is affected by the temperature of the air it travels in according to the following equation, Temperature is measured in Celsius, and velocity is measured in m/sec:

$$velocity = 331.3 + 0.61 \cdot Tc, \text{ where } Tc \text{ is the temprature of the air}$$

The code for the following problem can be found in *Figure 4.*

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    float velocity;
    int start,end;
    printf("Enter starting temp\n");
    scanf("%d",&start);
    printf("Enter ending temp\n");
    scanf("%d",&end);
    for(start; start<= end;start++){
        velocity= 331.3 + 0.61*start;
        printf("At %d degrees Celsius the velocity of sound
is %f\n",start,velocity);
    }
    return 0;
}
```

**Figure 4.** C code for problem 3.

In the following code user is prompted to enter the start temperature and the end temperature, and then we calculate the velocity of sound, in accordance with the given equation, as the temperature increases from the start temperature to the ending temperature. This was done using a for loop that looped over the starting temperature, slowly increasing it until it reached the value of the ending temperature, all while printing out the velocity at every step.

**8.4 - Interest Rate:** In the following question, the interest rate for an account is calculated given the account balance according to the rate schedules given in the question, the code for the following problem can be found in *Figure 5*.

```
#include <stdio.h>
float calculate_interest(float balance) {
    float interest = 0;
    if (balance <= 1000) {
        interest = balance * 0.015;
    } else {
        interest = 1000 * 0.015 + (balance - 1000) * 0.01;
    }
    return interest;
}
float calculate_minimum_payment(float total_due) {
    float minimum_payment = 0;
    if (total_due <= 10) {
        minimum_payment = total_due;
    } else {
        if ((total_due * 0.01) < 10) {
            minimum_payment = 10;
        } else {
            minimum_payment = total_due * 0.01;
        }
    }
    return minimum_payment;
}
int main(void) {
    float balance, interest, total_due, minimum_payment;
    int option;
    do {
        printf("Enter account balance:\n");
        scanf("%f", &balance);
        printf("Balance is: %.2f\n", balance);
        interest = calculate_interest(balance);
        printf("Interest is: %.2f\n", interest);
        total_due = balance + interest;
        printf("Total due is: %.2f\n", total_due);
        minimum_payment =
calculate_minimum_payment(total_due);
        printf("Minimum payment is: %.2f\n",
minimum_payment);
        printf("Choose one of the following options:\n");
        for (int i = 0;; ++i) {
            if (i > 0) {
                printf("Wrong input, Please try again\n");
            }
            printf("1- Calculate again\n");
            printf("2- Exit the program\n");
```

```
            scanf("%d", &option);
            if ((option == 1) || (option == 2)) {
                break;
            }
        }
    } while (option == 1);
    return 0;
}
```

**Figure 5.** C code for Problem 4.

The following code has 2 user-defined functions, the first one is `calculate_interest` which takes the user balance as input, and calculates the interest rate in accordance with the given rules, the second one is `calculate_minimum_payment` which takes the total due amount as input and returns the minimum payment in accordance with the given rules. In the end, the interest rate, minimum payment, and the total amount due are all printed the user is prompted for a rerun.

**8.5 - Integer Operations:** In the following question, we are given a positive integer and must check if it is a perfect square, reverse it, and return the sum of its digits. The code for the following problem can be found in *Figure 6.*

```
#include <iostream>
#include <cmath>
#include <string>
using namespace std;

void isPerfectSquare(int x){
    float squareNumber=sqrt(x);
    if((int)squareNumber==squareNumber){
      cout<<x<<" is a perfect square number.\n";
    }
    else{
      cout<<x<<" is not a perfect square number.\n";
    }
};

int reverseDigits(int num){
       int reversedNumber = 0;

    while (num != 0) {
        int digit = num % 10;
        reversedNumber = reversedNumber * 10 + digit;
        num /= 10;
    }

    return reversedNumber;

};
int sumDigits(int num){
  int sum=0;
  while (num != 0) {
        int digit = num % 10;
        sum +=digit;
        num /= 10;
    }

    return sum;
}
```

```
int main() {
  int number;
  cout<<"Enter your postive integer number : ";
  cin>>number;
  isPerfectSquare(number);

cout<<"Your reversed number "<<reverseDigits(number)<<"\n";
cout<<"Sum of digits of your number :
"<<sumDigits(number)<<"\n";
  return 0;
}
```

**Figure 6.** C++ code for problem 5.

The following code contains 3 user-defined functions, the first one is `isPerfectSquare` which checks if a number is a perfect sqaure by taking the square root of the number and checking to see if its floor is equal to its value, if so then it is a perfect square. The second function is `reverseDigits` which takes an integer input and defines a reversed integer, then we loop until the number is 0, as we loop we execlude the digit by taking the mod of the number to 10, then we add the number to reversed number multiplied by 10, and then we divide the number by 10. The third function is `sumsDigits` which works exactly the same as the second function except we don't multiply the value by 10.

**8.6 - Miles per Gallon:** In this question we are asked to calculate the miles per gallon ratio given the liters of fuel consumed and the distance travelled, where the relation between liters and gallons is given by the following equation:

$$Liters = 0.264179 \cdot n, \text{ where } n \text{ is the number of gallons.}$$

The code for the following problem can be found in *Figure 7*.

```
#include <stdio.h>

#define liter_to_gallon 0.264179

double calculate_miles_per_gallon(double liters_consumed,
double miles_travelled){
    double gallons_consumed = liters_consumed /
liter_to_gallon;

    return miles_travelled / gallons_consumed;
}

void main(){
    int option;
    while (1==1)
    {
        printf("Start calculation (1)\nQuit (0)\n");
        scanf("%d",&option);

        if (option == 0) exit(0);

        double liters_consumed;
        double miles_travelled;
        printf("Enter amount of gas consumed in liters: ");
```

```
        scanf("%lf",&liters_consumed);
        printf("Enter distance traveled in miles: ");
        scanf("%lf",&miles_travelled);
        printf("The consumption rate is : %0.3f
miles/gallon\n",
calculate_miles_per_gallon(liters_consumed,miles_travelled));
    }

}
```

**Figure 7.** C code for problem 6.

In the following code, we define a global value for the conversion rate between liters and gallons, the user if prompted for an action, after which the user is asked for the number of liters consumed and the miles travelled, in this code there is one user-defined function, it is `calculate_miles_per_gallon` the function takes the user given values as inputs, it firsts calculates the amount of gallons consumed according to the given equation, then it calculates the miles consumed per one gallon, then the calculated value is returned and printed.

**8.7 - Gravitational Attraction:** In this question we are asked to calculate the gravitational force between 2 objects; given their masses and the distance between them, the force is calculated according to the following equation:

$$F = \frac{G \cdot mass1 \cdot mass2}{d^2}, \; where \; G = 6.673 \cdot 10^{-8} \; \frac{cm^2}{g \cdot \sec^2}$$

The code for the following question can be found in *Figure 8.*

```
#include <iostream>
#include <cmath>
using namespace std;
#define UNIVERSAL_GRAVITATIONAL_FORCE 6.673e-8

double gravitationalForce(float massInkg1 , float
massInkg2,float distance){
  double result=
UNIVERSAL_GRAVITATIONAL_FORCE*(massInkg1*1000)*(massInkg2*100
0);
  double distanceInCm=distance*100;
  return result/pow(distanceInCm,2);
};
int main(){
    bool flag=true;
    float massInkg1,massInkg2,distance;
    string answer;
    while (flag)
    {
      cout<<"Enter the mass of first object in Kg : ";
      cin>>massInkg1;
      cout<<"Enter the mass of second object in Kg : ";
      cin>>massInkg2;
      cout<<"Enter the distance between two objects : ";
      cin>>distance;
      cout<<"The gravitational force is :
"<<gravitationalForce(massInkg1,massInkg2,distance)<<"\n";
      cout<<"Do you want to perform another calculation :
(y/n)";
```

```
      cin>> answer;
      if(answer=="y"){
        flag=true;
      }
      else{
        flag=false;
      }
    }


  return 0;
}
```

**Figure 8.** C++ code for Problem 7.

In the following code, there is one user-defined function, `gravitationalForce` which takes the masses of the bodies and the distance between them as input, then it calculates the upper term in the given equation then divides it by the lower term, the universal gravitational constant is defined as a global variable in the code, the user is prompted for the masses and the distance between them in kilograms and meters, respectively. During the calculations, the units are converted to fit the equation.

**8.8 - Bonus Problem 1:** In the following problem, we are asked to compare the values of 2 different pizzas given their diameters in inch, and their price, we are asked to find the more cost saving one, the code for the following problem can be found in *Figure 9.*

```
#include <stdio.h>
#include "math.h"
float calculateArea(float i) {
    double radius = i / 2;
    return M_PI * radius * radius;
}
float calculatePricePerSquaredInch(float i, float p) {
    return p / calculateArea(i);
}
int main(void) {
    float i1, i2, p1, p2;
    float smaller_pizza_price_per_squared_inch,
larger_pizza_price_per_squared_inch;
    printf("Enter the smaller pizza's diameter in inches\n");
    scanf("%f", &i1);
    printf("Enter the larger pizza's diameter in inches\n");
    scanf("%f", &i2);
    printf("Enter the smaller pizza's price\n");
    scanf("%f", &p1);
    printf("Enter the larger pizza's price\n");
    scanf("%f", &p2);
    smaller_pizza_price_per_squared_inch =
calculatePricePerSquaredInch(i1, p1);
    larger_pizza_price_per_squared_inch =
calculatePricePerSquaredInch(i2, p2);
    if (smaller_pizza_price_per_squared_inch <
larger_pizza_price_per_squared_inch) {
        printf("Smaller pizza is the best buy\n");
    } else if (larger_pizza_price_per_squared_inch <
smaller_pizza_price_per_squared_inch) {
        printf("Larger pizza is the best buy\n");
    } else {
```

```
            if (i1 < i2) {
                printf("Smaller pizza is the best buy\n");
            } else {
                printf("The 2 pizzas are identical\n");
            }
        }
        return 0;
}
```

**Figure 9.** C code for problem 8.

In the following code there are 2 user-defined functions, the first one is `calculateArea` which takes the diameter of the pizza as input, then calculates the area and returns the value. The second one is `calculatePricePerSquaredInch` which takes the price of the pizza and its area as inputs and returns the ratio between the price and area. The user is prompted to enter the area and price for both pizzas then the ratios are computed, the pizza with the least price per square inch is the better deal, if both are the same ratios, the smaller one is the better deal.

**8.9 - Bonus Problem 2:** In this question we are asked to implement the code for the bubble sort algorithm using functional programming, bubble sort works by comparing each 2 adjacent elements in array and swapping them if the one on the right is smaller than the one on the left, this sorting algorithm has $O(N^2)$ time complexity and O(1) space complexity, the code for the following problem can be found in *Figure 10.*

```
#include <stdio.h>

void bubble_sort(int size, int *array){
    for (int i = 0; i < size; i++){
        for (int j = 0; j<size-1; j++){
            if (array[j] > array[j+1]){
                int temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
            }
        }
    }
}
void main(){
    int array[] = {2,7,1,0,12,-2,5,24};
    printf("Array before sorting: ");
    for (int i = 0; i < sizeof(array)/sizeof(int); i++){
        printf("%d ",array[i]);
    }
    printf("\n");
    bubble_sort(sizeof(array)/sizeof(int), array);

    printf("Array after sorting: ");
    for (int i = 0; i < sizeof(array)/sizeof(int); i++){
        printf("%d ",array[i]);
    }
}
```

**Figure 10.** C code for Problem 9.

The following code has one user defined function, `bubble_sort` which takes as input the pointer to the first element of the array, and the size of the array. The size of the

array is computed by computing the size of the array in bytes and dividing it by the number of bytes reserved for integer variables.

## References


[1] Advantages and Disadvantages of IoT (Internet of Things).

[2] The Benefits of IoT: Real World Examples.

[3] Advantages and Disadvantages of IoT Explained – Clear Your Doubts.

[4] What is an IoT gateway?

[5] What is an IoT Gateway? - Check Point.

[6] What Is Fog Computing? Components, Examples, and Best Practices.

[7] Dynamically Controlling Offloading Thresholds in Fog Systems.

[8] What Is Cloud Computing? Definition, Benefits, Types, and Trends.

[9] An overview of HTTP.

[10] HTTP vs MQTT: Choose the Best Protocol for your IoT Project.

[11] Introducing the MQTT Protocol – MQTT Essentials: Part 1.

[12] 3 Types of Programming Errors and How to Avoid Them.

[13] Source code versus object code: key differences.

[14] Enum Types – Oracle.