Master's Thesis

# IoT Training System Using the Cooja Network Simulator

1710032 WANG Jidong

| | |
|---|---|
| Supervisor | Associate Professor Razvan Beuran |
| Main Examiner | Associate Professor Razvan Beuran |
| Examiners | Professor Yoichi Shinoda |
| | Professor Yasuo Tan |
| | Associate Professor Ken-ichi Chinen |

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

February 2019

**Abstract**

Opportunities, as well as challenges, always accompany the development of technology. Without a doubt, the Internet of Things (IoT), which is seen as another upcoming trend in technology after the Internet, is no exception. The development of IoT brings up a meaningful discussion about IoT security.

However, there is a variety of IoT devices, from the small sensor to the home router then to even big factory equipment. In our daily life, many of us may have several IoT devices but don't know they are IoT devices, let alone to manage these devices without being attacked. Therefore, IoT security education and training are extremely urgent.

This thesis presents IoTrain, which is an IoT training system using the Cooja network simulator. Users can select a tutorial by using the interface of IoTrain and complete various training under the guidance of the tutorial. These training includes viewing simulations and doing hands-on simulations in the Cooja network simulator.

The Cooja network simulator is an application included with Contiki OS and used as a tool in IoTrain. The Contiki OS is an operating system for low-power wireless IoT devices. The training content of IoTrain mainly involves wireless sensor networks (WSN), Routing Protocol for Low power and Lossy Networks (RPL), etc.

At the beginning of system development, the IoTrain was designed to meet the following requirements: open-source, parallel, for different level users, content-rich, low cost, and easy to manage. These system requirements are also indicators of the system evaluation.

Before designing the system, training mode, training process, and a structure of IoT training content were proposed.

The training mode and training process solves the problem of how to train users. Because the IoTrain will face many types of users if it is released, it is necessary to investigate and classify users first. In this thesis, users are roughly divided into three levels.

The first level is beginners. This group of people can be said to have no knowledge about IoT and IoT security, so their training should start from scratch.

The second level is intermediate users who have a vague understanding or a little knowledge of IoT and IoT security, but because of the lack of systematic training, these understanding and awareness have limited help in reducing their probability of being attacked in their daily lives.

The third level is advanced users who have considerable knowledge reserves, such as having studied relevant courses or have the ability to program. However, due to various restrictions, such as time or money costs, they did not get deeper into IoT and IoT security.

Referring to the way that most people usually learn, it is widespread to find a tutorial or material for self-study. Therefore, for all levels of users, especially beginners, tutorials will be presented to them as training content. Users can select the tutorials of interest through the interface and complete the corresponding training. This is the training process mentioned earlier.

When beginners become intermediate users, and then these intermediate users also complete the corresponding tutorials, they naturally become advanced users. At this stage, these advanced users will learn Contiki-based programming, application development, and even modify the source code of the Contiki OS to simulate some attacks under the guidance of advanced tutorials. From the "learning tutorials" to "viewing simulations" to "doing hands-on simulations", this series of training is summarized as the "learning-viewing-doing" training mode.

The training content is the core of the entire system. For a person who wants to get IoT training, the first thing is to learn some basic knowledge of the Internet of Things, and then to understand the advanced knowledge of IoT, such as network or security. Because from the basic to the advanced is the law of learning any knowledge. Thus, when designing the training content structure, the training content is first divided into three categories, namely system introduction, fundamental training, and security training.

IoTrain mainly consists of three parts, namely database, function, and interface.

The database is used to store the training content. Due to the entire system currently has only three training file types and the number of these files is small, therefore, the database is built without using professional database management software, and all files are classified and stored in folders and subfolders of the entire project.

Function includes configuring the system environment, displaying interface and options, as well as open tutorials and partial simulation files according to users' options.

The IoTrain interface is implemented as prompt options in the terminal window.

In the whole research, the most important contribution is the proposal of the training content structure. It is the blueprint of the development of IoTrain. Besides, it not only has a high reference value for the future development of the IoTrain system but also for the development of other IoT training systems.

The development of the IoTrain system based on the training content structure is the second contribution of this research. After proceeding the various stages, the system has reached the prototype stage, where the system's training content structure, functions, and interfaces have been implemented.

Using simulation in IoT training is indeed a useful attempt. The use of the Cooja network simulator not only reduces the cost of development but also increases the training content forms, while also making training more interesting and effective.

In this thesis, firstly, I will introduce the research background, motivation, and contribution. Secondly, I will add the background knowledge involved in the whole process of the research. Thirdly, I will introduce the IoTrain, including the system requirements, design, and implementation. Lastly, the system evaluation and conclusion will be presented.

**Keywords:** IoT, IoT security, IoT education and training, IoT simulation, Contiki, Cooja, WSN, RPL.

**Declaration:** I hereby declare that this whole dissertation is my own work and that it has not been previously included in any other thesis, dissertation or report.
**Student:** WANG Jidong

**Acknowledgements**

I want to express my gratitude to all those who helped me during the process of writing this thesis and during my study years in Jaist.

First and foremost, I want to extend my heartfelt gratitude to my supervisor, Associate Professor Razvan Beuran, whose guidance, valuable suggestions, and constant encouragement make me complete this thesis. His conscientious academic spirit and modest, open-minded personality inspire me both in academic study and daily life. He gives me much help and advice during the whole process of my writing, which has made my accomplishments possible.

Also, I would like to express my sincere gratitude to all the professors who have taught me in this university. Their instructions have helped broaden my horizon, and their enlightening teaching has provided me with a solid foundation to accomplish this paper and will always be of great value for my future career and academic research.

My thanks also go to the authors whose books and articles have inspired me in the writing of this paper.

Last but not least, my thanks would go to my beloved family for their thoughtful considerations and high confidence in me all through these years and their support without a word of complaint. I also owe my sincere gratitude to my friends and my classmates who have given me their help and their time in listening to me and helping me work out my problems during the challenging course of the thesis.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this chapter, I will first introduce the relevant background of IoT security, which leads to the research motivation. Then I will present the contribution of this thesis and some necessary introductions about the Contiki operating system and the Cooja network simulator.

## 1.1 Background

Since the advent of the Internet, the number and variety of devices connected to the Internet have been increasing. After the emergence of the concept of the Internet of Things, the number and type of the "Things" is like a blowout, exponential growth.

One application of the IoT is to embed sensors into a variety of things which integrated into the Internet. With this kind of use, people can remotely acquire and exchange data, achieve interaction with other people or equipment, and thus promote the relationship between human and the environment. According to [1], by the end of 2020, it is said that there would be around 30 billion connected devices and then may be nearly triple against 80 billion five years later. Furthermore, the total digital data created globally will increase from 4.4 zettabytes in 2013 to 44 zettabytes in 2020, and by 2025 it is likely to reach 180 zettabytes.

Users are data, and data is the value. In the face of the immense amount of value, many hackers are already in action to abuse their talents every way they can. The most recent and famous IoT attack is Mirai which infected a large number of IoT devices, resulting in the inaccessibility of several high-profile websites such as GitHub, Twitter, Reddit, Netflix, Airbnb, and many others [2]. Therefore, in the era of the IoT, where there are a massive number of devices and large amounts of data, the security issue is one of the most

critical issues.

However, explosive devices, insecure IoT communication, and users who lack security knowledge and awareness are constantly giving hackers chances. A report of HP reveals that 70% of IoT devices are vulnerable to attacks, and 90% of devices collected at least one piece of personal information via the device, the cloud, or its mobile application [3]. In IoT-type system, the lack of secured links exposes data to attacks and theft. Secure communication still needs the multi-level configuration and application-level proprietary algorithms, which discourages users from implementing protection [4]. And the study [5] suggests that half of the Internet-connected users based in the US have not heard of the concept of "Internet of Things," which means that the situation in other countries may be worse.

For the issues of explosive device growth, Edith Ramirez shared three key steps [6] that companies should take to enhance the protection of consumers' security and privacy to build consumers' trust in devices:

1. Adopting "security by design"
2. Engaging in data minimization
3. Increasing transparency and providing consumers with notice and choice for unexpected data use

For the issues of insecure IoT communication, though there is still a long way to go, key technologies shown by Xu in [7] such as certification and access control, data encryption, middleware, and cloud computing, are considered technologies that promise to improve the security status.

## 1.2 Motivation

Professionals keep working hard to create the IoT society as safe as possible. However, not everyone is a professional. Users often lack security awareness and necessary knowledge, and most of the attacks are formed because hackers discovered the users' negligence. In my point of view, providing users with IoT training is the most effective way to reduce attacks.

The current available IoT training systems are handful, and most of them are lack of content and using cyber training's "Question–Answer–Test" mode, which is unattractive. This situation gave me the motivation to develop a free, attractive and content-rich IoT training system, which was later named as IoTrain.

## 1.3 Contribution

The following points of this thesis can be considered as contributions:

1. On the basis of considering the types of IoT devices, training objects and training methods, a new training content structure is proposed.

2. Based on the Cooja network simulator, an IoT training system was developed, which consists of the following components:

   - Database
   - Functions
   - Interface

3. According to the training content structure, tutorials and simulations were made and added into IoTrain's database as the training content. The tutorial guides the user training, and the simulation increases the fun of training.

# Chapter 2

# Background

To better understand and conduct this research, a knowledge framework consisting of the following five sections, which are IoT, wireless sensor network, RPL, Tools, has been built. This chapter will introduce these five sections.

## 2.1 Internet of Things (IoT)

### 2.1.1 Definitions

What is the Internet of Things? How is it defined? Many professionals and organizations have given their explanations. For example, in Wikipedia, the Internet of Things is defined as "the network of devices such as vehicles, and home appliances that contain electronics, software, actuators, and connectivity which allows these things to connect, interact and exchange data" [11], while in [12], it is more professionally defined as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies". The definition in [13], "a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction", is closer to my understanding of IoT.

### 2.1.2 Architectures

Due to the different understanding and definition of IoT, correspondingly, the IoT architecture has been proposed in many different types (shown in

Figure 2.1). One famous mode is three-layer architecture [14], including Application, Network and Perception layers. In [12], a four-layer reference mode was proposed, consisting of Application layer, Service support and Application support layer, Network layer and Device layer. Another architecture I want to mentioned here is composed of five layers [15]: a Business layer, an Application layer, a Processing layer, a Transport layer, and a Perception layer.

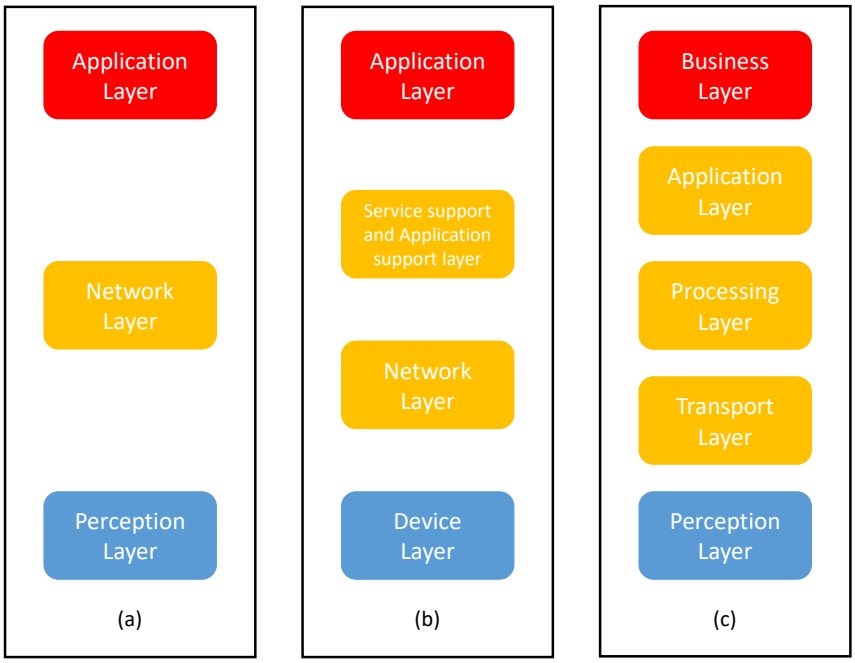


Figure 2.1: IoT architecture: (a) Three-layer. (b) Four-layer. (c) Five-layer.

### 2.1.3 Elements

Identification, sensing, communication, computation, services, and semantics [16] proposed by Al-Fuqaha as the six main elements that can represent the functionality of IoT. Figure 2.2 below shows these elements that help to understand, and Figure 2.3 shows the technology samples in each element block.

### 2.1.4 Common Standards

To simplify the development and accelerate the popularity of IoT, many organizations have developed various standards. Engineers can collaborate

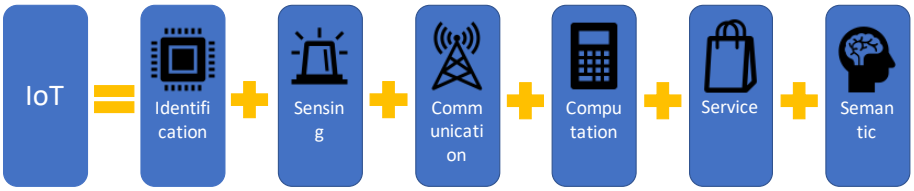


Figure 2.2: IoT elements

| IoT Elements | | Samples |
|---|---|---|
| **Identification** | **Naming** | EPC, uCode |
| | **Addressing** | IPv4, IPv6 |
| **Sensing** | | Smart Sensors, Wearable sensing devices, Embedded sensors, Actuators, RFID tag |
| **Communication** | | RFID, NFC, UWB, Bluetooth, BLE, IEEE 802.15.4, Z-Wave, WiFi, WiFiDirect, LTE-A |
| **Computation** | **Hardware** | SmartThings, Arduino, Phidgets, Intel Galileo, Raspberry Pi, Gadgeteer, Z1, Tmote Sky |
| | **Software** | OS(Contiki, TinyOS, LiteOS, Riot OS, FreeRTOS, Android);<br>Cloud(Nimbits, Hadoop, etc.) |
| **Service** | | Identity-related (shipping), Information Aggregation (smart grid), Collaborative-Aware (smart home), Ubiquitous (smart city) |
| **Semantic** | | RDF, OWL, EXI |

Figure 2.3: IoT elements and technology samples

and develop universal products according to the same standards, even if they belong to different groups or departments. Some common standards [16] are shown in figure 2.4.

<table>
<tr><td colspan="2">Application Protocol</td><td>DDS</td><td>CoAP</td><td>AMQP</td><td>MQTT</td><td>MQTT-SN</td><td>XMPP</td><td>HTTP REST</td></tr>
<tr><td colspan="2">Service Discovery</td><td colspan="3">mDNS</td><td colspan="4">DNS-SD</td></tr>
<tr><td rowspan="4">Infrastructure Protocols</td><td>Routing Protocol</td><td colspan="7">RPL</td></tr>
<tr><td>Network Layer</td><td colspan="3">6LoWPAN</td><td colspan="4">IPv4/IPv6</td></tr>
<tr><td>Link Layer</td><td colspan="7">IEEE 802.15.4</td></tr>
<tr><td>Physical/Device layer</td><td colspan="2">LTE-A</td><td colspan="2">EPCglobal</td><td colspan="2">IEEE 802.15.4</td><td>Z-Wave</td></tr>
<tr><td colspan="2">Influential Protocols</td><td colspan="3">IEEE 1888.3, IPSec</td><td colspan="4">IEEE 1905.1</td></tr>
</table>

Figure 2.4: IoT common standards

### 2.1.5 Applications

The Internet of Things has many application domains, and due to its rapid development, the number of domains is increasing. Figure 2.5 [17] shows the top 10 IoT segments in 2018.

### 2.1.6 Challenges and Future Directions

In [18], Yogita Pundir, M., Sharma, M. N., and Singh, Y. proposed some challenges facing the development of IoT, as follows:

1. Standards and interoperability
2. Security
3. Trust and privacy
4. Complexity, confusion and integration issues
5. Evolving architectures, protocol wars and competing standards
6. Concrete use cases and compelling value propositions

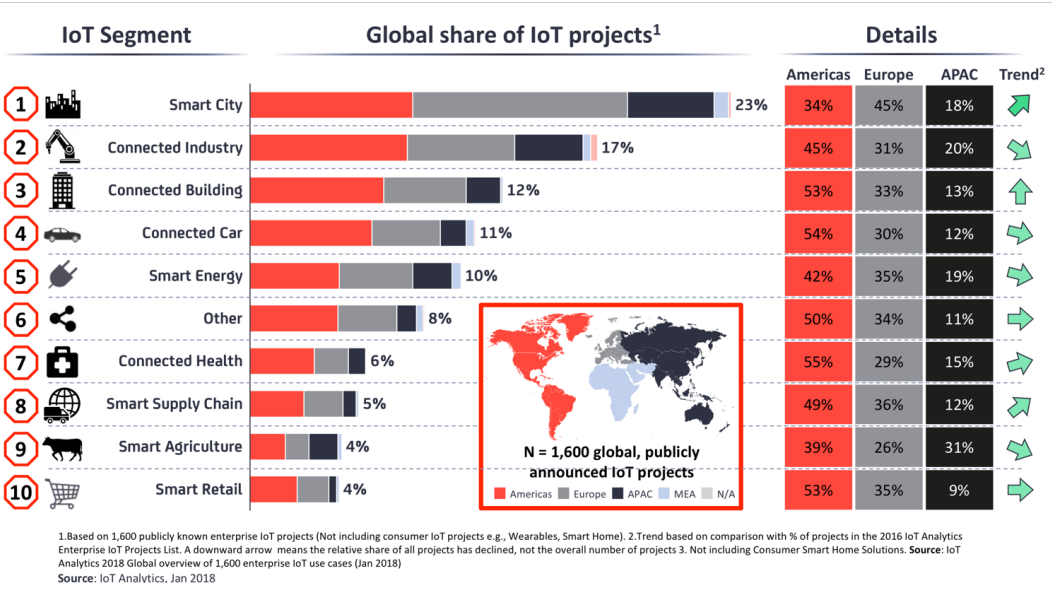


Figure 2.5: Top 10 IoT segments in 2018 [17]

It can be seen that the security issues raised at the beginning of this thesis are also included.

## 2.2 Wireless Sensor Network (WSN)

### 2.2.1 What is a WSN?

A wireless sensor network (WSN) consists of many wireless sensor nodes or motes. These nodes/motes are usually low-powered, storage capacity and processing constraint and working in small distances. Every node/mote is mainly composed of the following six parts:

- **Embedded processor**: low-powered, processing small tasks, and providing the nodes/motes with management functions.
- **Sensors**: sensing physical phenomena such as light, heat, pressure, etc. Due to bandwidth and power constraints, nodes/motes primarily support low data units with limited computational power and a limited sensing rate.
- **Memory**: storing programs (instructions executed by the processor) and data (raw and processed sensor measurements).

- **Transceiver**: transmitting and receiving data wirelessly, at a low frequency and over a short range.
- **Power source**: rechargeable batteries are usually used.
- **Operating Systems**: Contiki OS, Tiny OS, FreeRTOS are the examples of operating systems that are used for WSNs.

As a method of information collection, wireless sensor networks build information and communication systems which significantly improve the reliability and efficiency of IoT. Compared to wired solutions, devices of WSNs are more flexible and more comfortable to develop. With the rapid development of sensor technology, WSN now has been the key technology of IoT.

### 2.2.2 Protocol Stack for WSN

Figure 2.6 shows the WSN protocol stack used by the sink and all sensor nodes/motes [19].

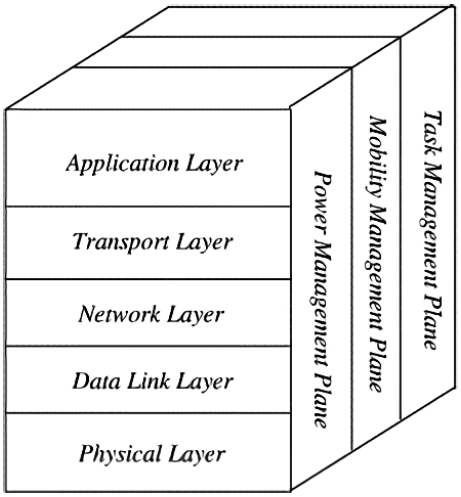


Figure 2.6: Protocol stack for WSN

As shown, the protocol stack consists of six layers and is divided into three management planes.

**Physical Layer** is designed for dealing with the frequency selection, frequency generation, modulation, signal detection, and encryption.

**Data Link Layer** is designed for managing the error control and the Medium Access. This layer ensures reliable end-to-end connections in a communication network.

**Network Layer** is in charge with routing the data provided by the transport layer.

**Transport Layer** is responsible for maintaining the flow of data to keep WSN operable when needed.

**Application layer** provides the communication interface.

**Power Management Plane** manages how sensor nodes use their power and it also determines the power consumption rates of the sensing, computing, and communicating.

**Mobility Management Plane** detects the movement of the sensor nodes/motes. It also records the mobility of the sensor nodes. Therefore, the route back to the user is always maintained, and nodes/motes can manage their power to complete tasks by considering this situation.

**Task Management Plane** organizes the specific area's sensing and detecting events, which means in the same area, not all sensor nodes/motes are performing the sensing tasks simultaneously.

## 2.3 RPL: IPv6 Routing Protocol for Low Power and Lossy Networks

### 2.3.1 Low Power and Lossy Network (LLN)

Low Power and Lossy Network(LLN) is a kind of network that composed of many embedded devices with limited power, memory, and processing resources. These devices are interconnected by a variety of links, such as IEEE 802.15.4 or low-power Wi-Fi.

LLN has a broad scope of application areas including monitoring, building automation, connected home, health care, environmental monitoring, urban sensor networks, energy management, assets tracking, and refrigeration. The WSN reviewed earlier is a particular type of LLN.

### 2.3.2 RPL Overview

RPL is a distance vector and source routing protocol for LLNs using IPv6 addressing.

#### Distance Vector

The term distance vector refers to the fact that the protocol manipulates vectors of distances to other nodes in the network. This protocol is based on calculating the direction and distance to any link in a network. "Direction"

means the next hop address and the exit interface, and "distance" means a measure of the cost to reach a particular node.

The protocol stipulates that the least cost route (best path) between any two nodes is the route with minimum distance. The cost of reaching a destination is calculated using various route metrics. Therefore, to efficiently find the lowest cost route, each node maintains a vector (table) of minimum distance to every node.

### Source Routing

The source routing in [20] is defined as allow a sender of a packet to partially or wholly specify the route the packet takes through the network.

### IPv6 Control Messages

The RPL specification defines four types of IPv6 control messages for topology maintenance and information exchange (Figure 2.9). These control messages are:

- **DODAG Information Object (DIO)**: The primary source of routing control information. Storing information like the current rank of a node, the current RPL Instance, the IPv6 address of the root, etc.
- **Destination Advertisement Object (DAO)**: Used to advertise information required to support downward traffic towards leaf nodes.
- **DODAG Information Solicitation (DIS)**: Used by nodes to request graph related information from the neighboring nodes.
- **Destination Advertisement Object Acknowledgement (DAO-ACK)**: Sent by a DAO recipient in response to a DAO message.

### RPL Construction Process

RPL organizes the topology of an LLN's nodes into a Directed Acyclic Graph (DAG) (Figure 2.7) where the default routes among the LLN's nodes are kept. A DAG is a finite directed graph with no directed cycles. Each node in the DAG is assigned a rank, which is calculated by a function called the Objective Function (OF). The rank monotonically decreases from the bottom (the DAG root has the lowest rank) and defines the node's position relative to other nodes concerning DAG root (shown in figure 2.9).

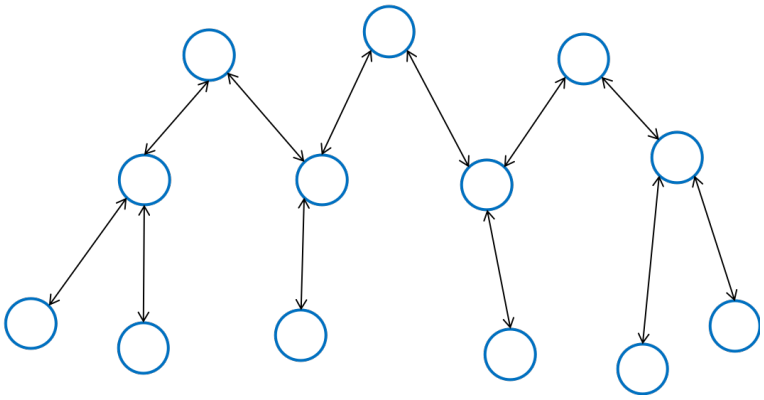

Figure 2.7: Directed acyclic graph (DAG)

The DAG can be partitioned into one Destination Oriented Directed Acyclic Graph (DODAG) (Figure 2.8) or more. That is, a DODAG consists of a root node inside the DAG, also called DAG root, which performs the function of a data sink or a gateway. One or more DODAGs can form an RPL instance. The DODAGs in one RPL instance share a unique ID called RPLInstanceID (Figure 2.9).

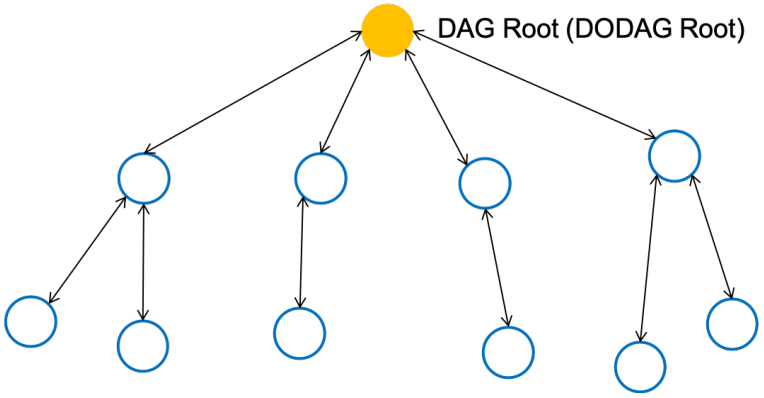


Figure 2.8: Destination oriented directed acyclic graph (DODAG)

To identify and maintain the topology, RPL utilizes four central values attached within RPL control messages (Figure 2.9). Rank and RPLInstanceID are two of them. The other two are DODAGID and DODAGVersionNumber. DODAGID is to identify a DODAG. The combination of a DODAGID and an RPLInstanceId can uniquely represent a DODAG. The DODAGVersion is a specific iteration of a DODAG with a given DODAGID, and the DODAGVersionNumber is a sequential counter that is incremented by the root to form a new version.

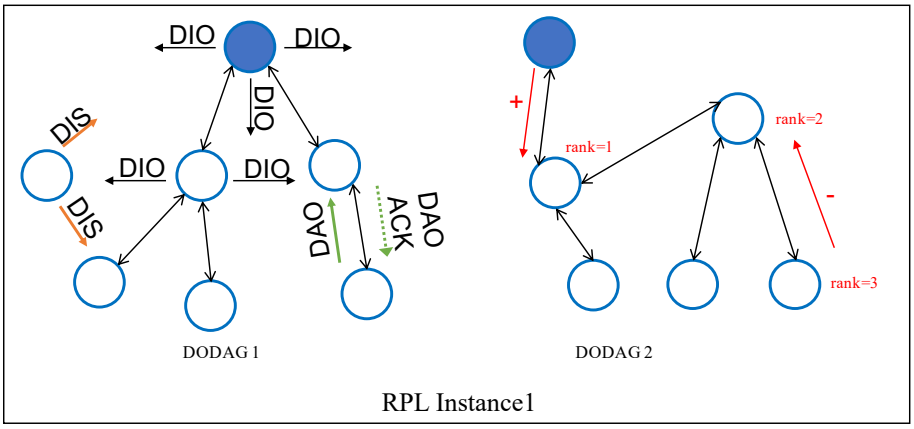


Figure 2.9: An RPL instance

## 2.4 Attacks in RPL-based IoT

RPL overcomes the routing problems in LLN. It implements a way for energy consumption saving, such as controlling the dynamic sending rate of messages, and resolving the issue of topological inconsistency only when data packets have to be sent. By using IPv6, it not only supports upward transmission but also flows from a gateway node to all other nodes.

However, RPL is also exposed to a variety of attacks. The severe consequences are particularly evident regarding network performance and resources. In [21], taxonomy has been proposed that divides all RPL-against attacks into three categories. The figure 2.10 shows the taxonomy.

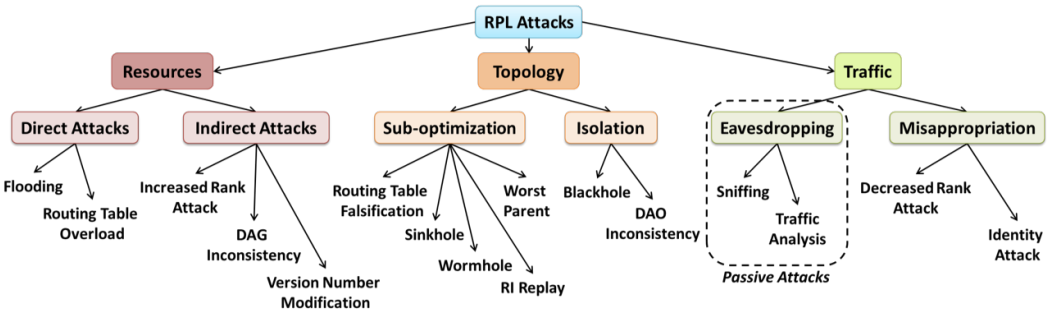


Figure 2.10: Taxonomy of attacks against RPL networks

### 2.4.1 Attacks on Resources

Attacks on resources involve the exhaustion of network resources, which means that the purpose of malicious nodes is to overload energy, memory or/and power consumption. This kind of attack can be achieved by forcing legitimate nodes to perform unnecessary operations to increase the use of their resources and may affect the availability of the network by blocking the available links or disabling the nodes, and may, therefore, change the lifetime of the network.

This category can be further subdivided into two sub-categories: direct attacks, in which malicious nodes directly generate overloads by interfering with the network; indirect attacks, in which malicious nodes interfere with other nodes to be overloaded.

### 2.4.2 Attacks on Topology

Attacks on Topology include attacks against RPL network topology. These attacks aim to disrupt the normal operation of the network. These may lead to the isolation of one or more nodes. This category can also be divided into two sub-categories: sub-optimization, which means that the network will converge to the non-optimal form, resulting in poor performance; isolation means isolating nodes or subsets of nodes, cutting them off from the rest of the network, thereby cutting off the root node.

### 2.4.3 Attacks on Traffic

Attacks on Traffic include attacks on network traffic. These attacks aim at introducing malicious nodes into the network rather than interfering with its work. This can lead to information leakage by eavesdropping the traffic or impersonating legitimate nodes. This category is further divided into two subcategories: eavesdropping (passively) information forwarded through the network; misappropriating a node or a group of nodes, that is, tampering with legitimately exchanged information.

## 2.5 Tools Introduction

This section mainly introduces the tools used for building the research environment. The whole research environment is shown below (Figure 2.11).

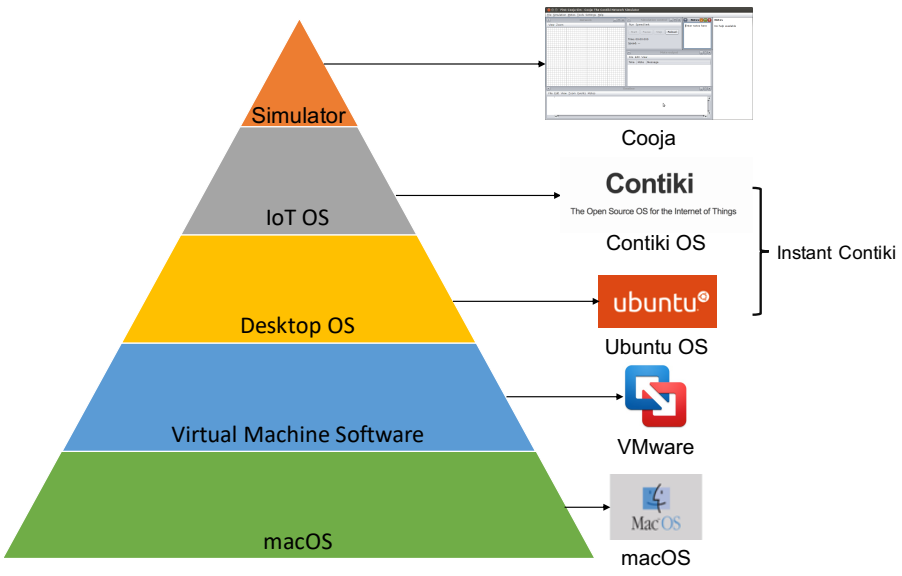


Figure 2.11: Research tools

### 2.5.1 Contiki Operating System

Contiki is an open source operating system for the Internet of Things, which connects tiny, low-cost, low-power microcontrollers to the Internet. Contiki is a powerful toolbox for building complex wireless systems and supports fully standard IPv6 and IPv4, along with the recent low-power wireless standards: 6lowpan, RPL, CoAP [8].

Contiki is an event-driven system in which processes are implemented as run-to-complete event handlers. The Contiki system consists of two parts: the system core and the loaded program. The core includes the Contiki kernel, the program loader, the language runtime, and the communication stack with communication hardware device drivers. The program loader loads the program into memory, which can be obtained from the host using a communication stack or from an additional storage device.

Contiki applications are written in standard C. With the Cooja network simulator, Contiki networks can be emulated before burned into hardware. As there are plenty of examples in the Contiki source code tree to help users get started with their code, and most have a corresponding Cooja simulation available, Contiki OS & the Cooja network simulator are very suitable as tools for IoT training system.

### 2.5.2 The Cooja Network Simulator

The Cooja network simulator is an extensible Java-based simulator capable of emulating Tmote Sky, Z1 or other nodes. It provides a simulation environment that allows developers to both see their applications run in large-scale networks or extreme detail on fully emulated hardware devices.

In Cooja's simulation environment, the code to be executed by the node is the same firmware that will be uploaded to the physical node, and all the interactions with the simulated nodes are performed via plugins like Simulation Visualizer, Timeline, Collect-view, and Radio logger.

The Cooja network simulator can store the simulation in an XML file with extension "CSC", which means "Cooja Simulation Configuration". This file contains information about the simulation environment, plugin, the nodes, and its positions, random seed and radio medium, etc.

### 2.5.3 Instant Contiki

Instant Contiki (Figure 2.12) is an entire Contiki development environment. It is an Ubuntu Linux virtual machine that runs in VMWare player and has Contiki and all the development tools, compilers, and simulators used in Contiki development installed [9]. Considering that it uses virtual machines, it is ideal for fast, large-scale configuration of training environments.

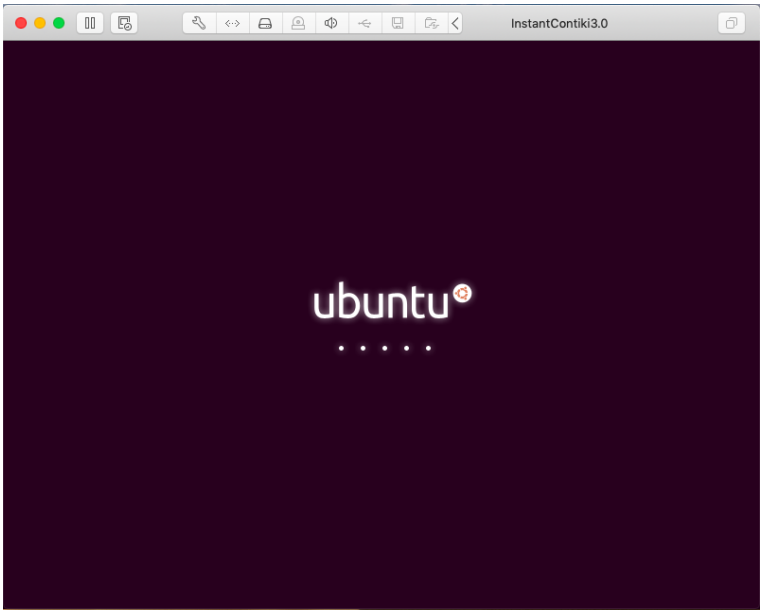


Figure 2.12: Instant Contiki

# Chapter 3

# IoTrain: IoT Training System

This chapter is the core of the entire research. Firstly, I will introduce the requirements for the system at the beginning of the study. Secondly, I will present the design and implementation of the training content structure. Thirdly, I will introduce the design and implementation of the system structure. The figure below shows an overview of the IoTrain.

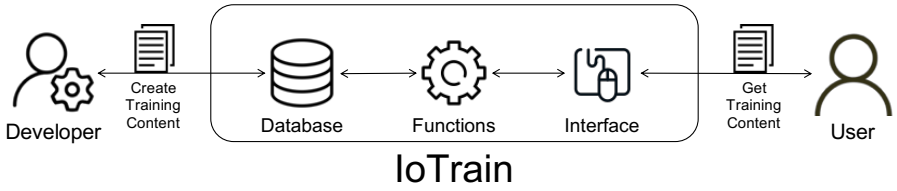


Figure 3.1: Overview of IoTrain

## 3.1 System Requirements

At the beginning of system development, the IoTrain was designed to meet the following requirements:

- **Open-source**
  As Pham Duy Cuong suggested in [10], having the IoT training system as an open-source product is also the best way to do a large-scale program that reaches different levels of users ranging from top organizations, companies to young people in universities, colleges, and even high schools.

- **Parallel**
  The system should be designed to serve multiple users simultaneously.

- **For different level users**
  As different users have different levels of knowledge about IoT, the system should be designed for different level users, from beginner to advanced.

- **Content-rich**
  Compared with the Internet, the IoT involves more hardware, so the content and mode of IoT training should be more abundant than Cyber Training. Therefore, the system should be designed to allow users to access as much hardware as possible by multiple training modes.

- **Low cost**
  For users, the low cost means that the installation and use of the system should be easy, the content acquisition should be free, and the training should be efficient. And for developers, due to the IoT training requires physical hardware, most of which are expensive, so the low cost here means that the cost of equipment should be minimized, especially for individual developers and small team developers.

- **Easy to manage**
  Management objects refer to the system structure and the training content. They should be reasonably designed from the outset, which not only improves the efficiency of development and use but also reduces the cost.

If the system requirements can be completed and widely used, it will not only help to improve people's security awareness and knowledge of IoT but also reduce the occurrence of IoT attacks.

## 3.2 System Design and Implementation

This section can be further divided into three subsections. The first subsection introduces the training mode and process. The second subsection introduces the training content structure design that includes the fundamental training and security training. The third subsection introduces the model of the system structure design, including the database design and the interface design.

### 3.2.1 Training Mode and Process

"How to train users" is an important issue after the system requirements have been proposed. This issue can be subdivided into issues about training

mode and training processes.

### User Level Classification

Because the IoTrain will face many types of users if it is released, it is necessary to research and classify users first. In this thesis, they are roughly divided into three levels (Shown in Figure 3.2).

The first level is beginners. This group of people can be said to have no knowledge about IoT and IoT security, so their training should start from scratch.

The second level is intermediate users who have already had a vague understanding or a little knowledge of IoT and IoT security, but because of the lack of systematic training, these understanding and awareness have limited help in reducing their probability of being attacked in their daily lives.

The third level is advanced users who have considerable knowledge reserves, such as having studied relevant courses or have the ability to program. However, due to various restrictions, such as time or money costs, they did not learn more about IoT and IoT security.

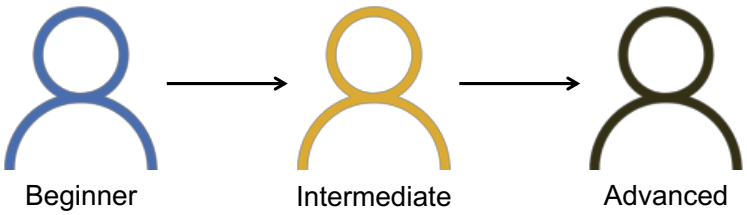


Figure 3.2: User levels

### Training Mode and Process

After classifying users, the next step is to adopt different training methods for different levels of users. Referring to the way most people usually learn, it is widespread to find a tutorial or material for self-study. Therefore, for all levels of users, especially beginners, tutorials will be presented to them as training content. Users can select the tutorials of interest through the interface and complete the corresponding training.

Users will learn to use the Cooja simulator under the guidance of the tutorial and then view the simulation to deepen their understanding of the tutorial content. Viewing the simulations is for most intermediate users, so they can also skip some tutorials learning and start directly from this step.

When beginners become intermediate users, and then these intermediate users also complete the corresponding tutorials, they naturally become advanced users. At this stage, these advanced users will learn Contiki-based programming, application development, and even modify the source code of the Contiki OS to simulate some attacks under the guidance of advanced tutorials.

From the "learning tutorials" to "viewing simulations" to "doing hands-on simulations", this series of training is the "learning-viewing-doing" training mode proposed in this thesis. Following this mode, the training process was designed and shown in Figure 3.3.

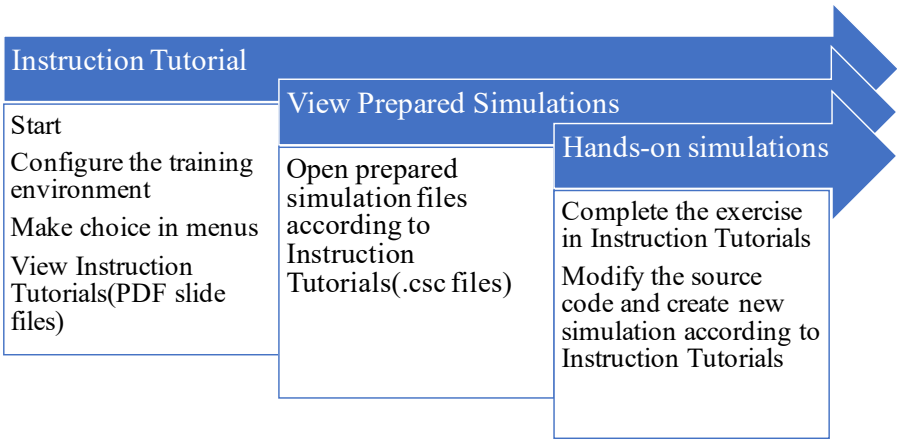


Figure 3.3: Training process

### 3.2.2 Training Content Structure Design

The training content structure is the core of the entire system. For a person who wants to get IoT training, the first thing is to learn some basic knowledge of the Internet of Things, and then to understand the advanced knowledge of IoT, such as network or security. Because from the basic to the advanced is the law of learning any knowledge. Thus, when designing the training content structure, the training content is first divided into three categories, namely system introduction, fundamental training, and security training, as shown in the following figure (Figure 3.4). It is worth mentioning that all current training content has been created at the time of writing this thesis, and the creation process will be explained in the next section. The training content structure may be adjusted in subsequent developments.

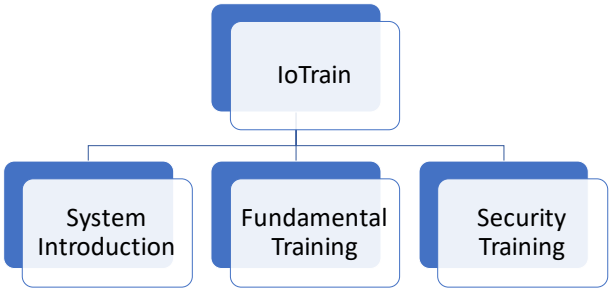


Figure 3.4: Training content

The system introduction is the beginning training content of the whole system, which is aimed at all levels of users. Its content covers the background of IoT and IoT security and how to use the system.

The fundamental training is further subdivided into two sub-categories: training with single node and training with networks. In the first sub-category, the basic of Contiki OS and the Cooja network simulator, as well as some Contiki-based IoT devices, such as actuators, controllers, and sensors, will be introduced. The second sub-category will focus on some communication methods in the network. The content structure of this part is shown below (Figure 3.5).

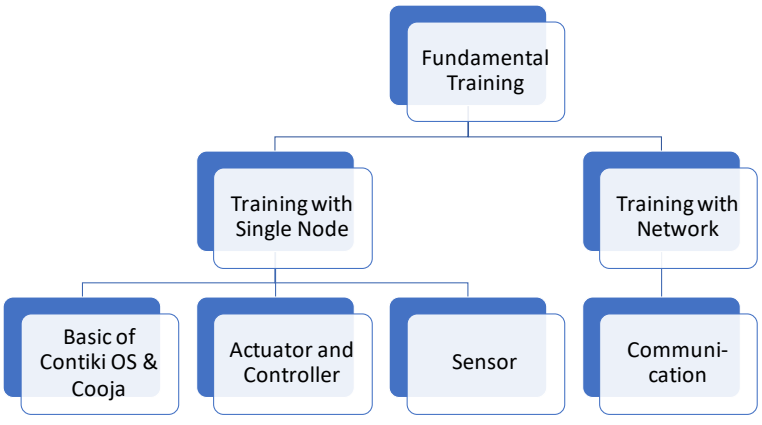


Figure 3.5: Fundamental training content

The security training content structure is mainly designed according to [21], but at present only some of the attack simulations are implemented, so only

the corresponding tutorials have been made for the implemented attacks. The content structure of this part is shown below (Figure 3.6).

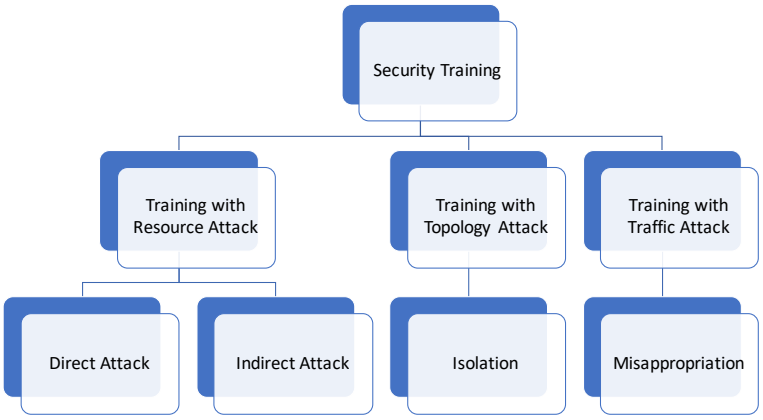


Figure 3.6: Security training content

### 3.2.3 Training Content Creation

#### Overview

The training content Creation refers to adding specific executable files to the bottom classification blocks after the design of training content structure, which can be directly invoked by users through the system interface.

There are three types of these files. The first is the Portable Document Format (PDF) file, which is used as training tutorials. The second is the Cooja Simulation Configuration (CSC) file, which can be imported into the Cooja network simulator and then become the object of the "viewing" mentioned in the section 3.2.1. The third is the C file, which is the application source code developed by the advanced users under the guidance of the tutorials or the source code files of the Contiki that needs to be modified to implement the attacks.

The following figure shows a content creation map of the entire system (Figure 3.7). In fact, except for the system introduction part, which only contains PDF files, the rest of the training content contains the three file types mentioned above.

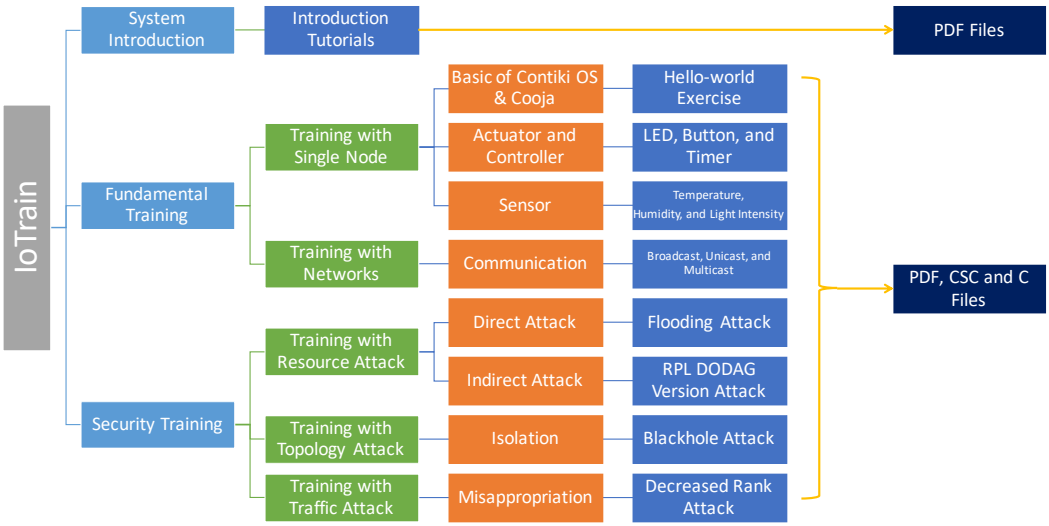


Figure 3.7: Content creation map

### Tutorials Creation

To create tutorials, developers first need to collect and learn a variety of materials, and then use the Microsoft Powerpoint software to make slides and export them as PDF files, and finally store these pdf files in the database. The creation procedure and a completed tutorial example are shown below (Figure 3.8 and Figure 3.9).

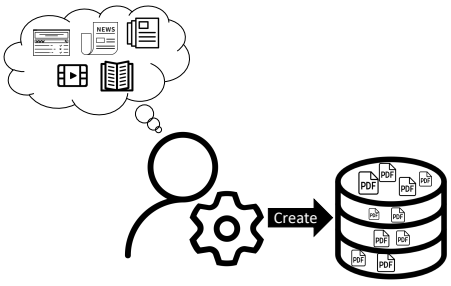


Figure 3.8: Tutorials creation

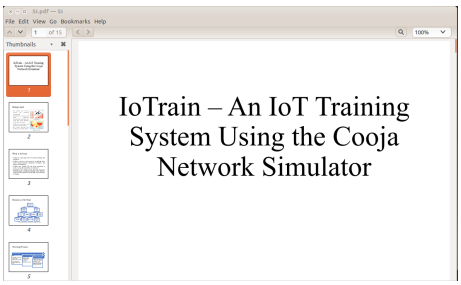


Figure 3.9: Tutorial example

### Simulation Implementation

The simulation implementation is divided into two categories according to the content, one is the implementation of fundamental training, and the other is the implementation of security training.

To implement a fundamental training simulation, firstly, according to the programming rules of Contiki OS, write a Contiki application using C language and save it in a C file. Then import the application into the Cooja network simulator, and select an appropriate hardware platform to compile and generate the simulation. Finally, save the simulation in the database as a CSC file, so that the user can open the view through the interface. It should be noted that in a simulation, each source file can only generate one kind of mote, but the number is not limited. The implementation procedure is shown below (Figure 3.10).

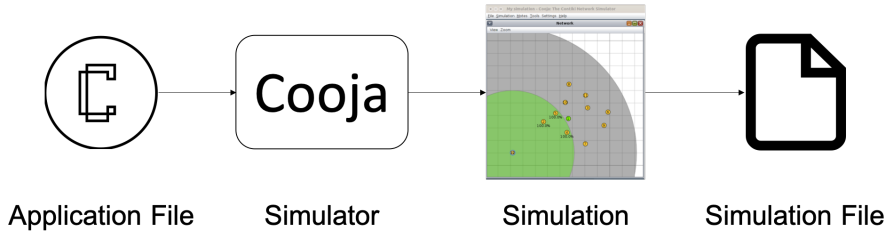


Figure 3.10: Fundamental training simulation implementation

As for the implementation of the security training simulation, there will be more steps. Before introducing these steps, an application called collect-view (Figure 3.11) will be introduced, which is used for all security training simulations.

Collect-view is a Java-based application in Contiki OS for sensor visualization, usually used in the Cooja network simulator. It involves a mote acting as a sink and other motes acting as sources. In the simulation, when sources send important parameters to sink, collect-view will be used to visualize these parameters in a graphical user interface. In the attack simulation, it will be used to observe the impact of malicious nodes on the network.

As shown in Figure 3.7, four attacks have been implemented. Each kind of attack has two simulations, one is a reference simulation, which means a normal sensor network simulation without malicious motes. The other is attack simulation. The attack simulation is to replace a source mote in the reference simulation with a malicious node, which will perform some kind of attack. Therefore, to create an attack simulation, a reference simulation should be created first.

The C files that implement the security training simulations are all using the sample files already in Contiki OS. The sink mote's path is "*contiki/examples/ipv6/rpl-collect/sink.c*", and source mote's path is "*contiki/examples/ipv6/rpl-collect/udp-sender.c*". In the Cooja network simulator, import and compile the sink.c file to generate a sink mote, then import and

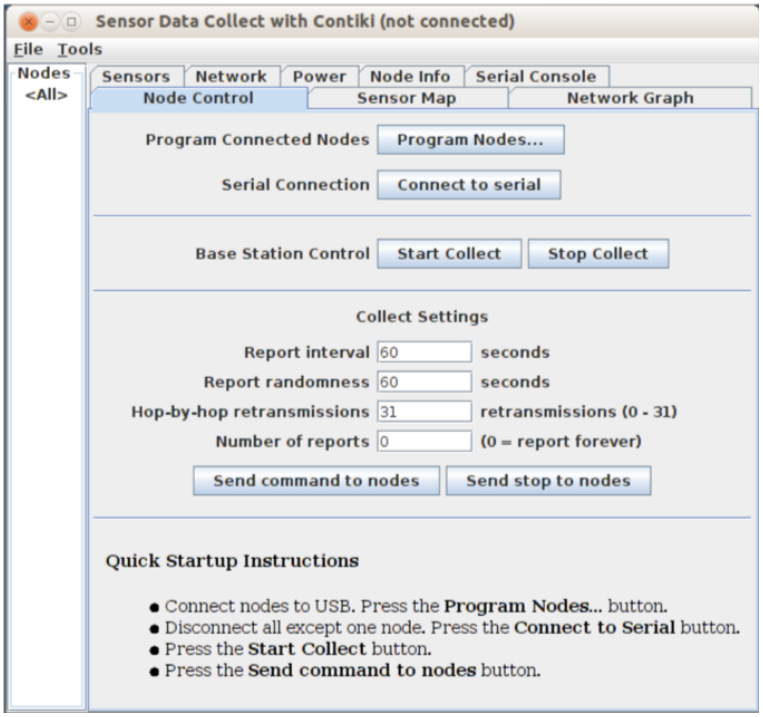


Figure 3.11: Collect-view

compile the udp-sender.c file to generate multiple source motes. All these motes make up a reference simulation of a wireless sensor network. Essentially, the steps to create a reference simulation are the same as the steps to create a fundamental training simulation. The differences are in creating attack simulations.

The attack simulation can be done by the following steps:

1. Duplicate the Contiki OS folder to create a new Contiki OS instance.

2. Modify the correspondent files according to the attack. For example, to implement a flooding attack, the file rpl_private.h and the file rpl_timers.c need be modified.

3. Create a new mote (malicious) in Cooja by compiling the udp-sender.c file within the new Contiki instance.

4. Add the mote to the reference simulation.

The simulation files will also be saved in the database as CSC files. Intermediate users will view these simulations to gain insight into these attacks,

while advanced users will modify Contiki's source code to implement these attacks themselves under the guidance of tutorials.

The reference simulation (Figure 3.12) and attack simulation (Figure 3.13) of the implemented flooding attack will be given below. The other three types of attacks are similar and are not described here.

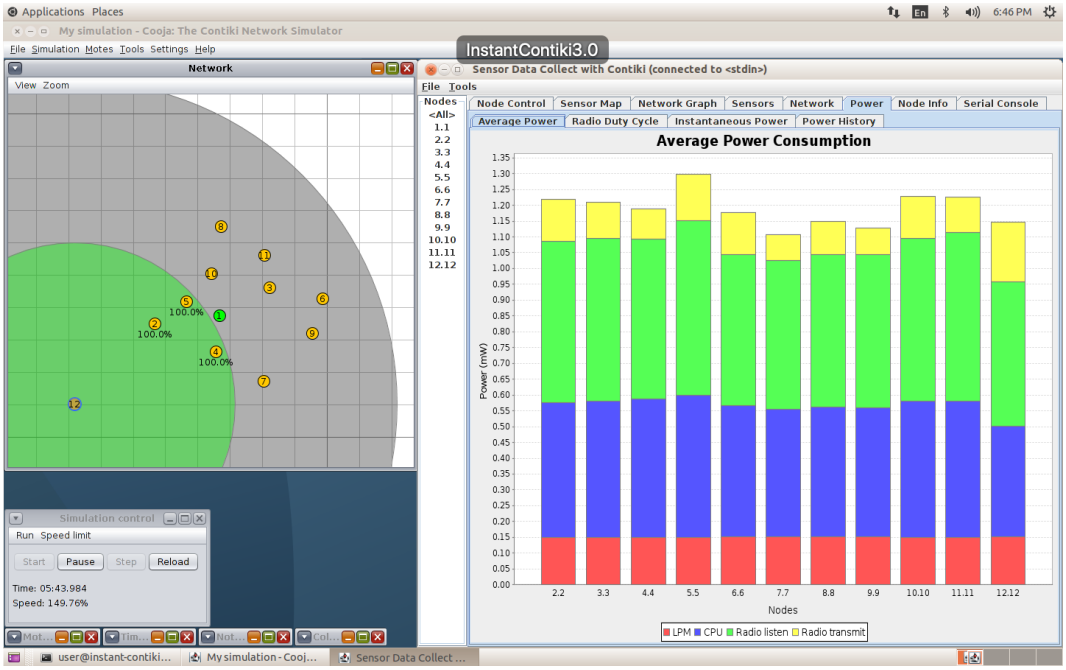


Figure 3.12: Reference attack of flooding attack

In the collect-view of the reference simulation:

- Node 1 in color green is a sink node which acts as a border router
- Other nodes in color yellow are sender node which act as sensors
- Node 2,4 and 5 are in the range of node 12
- All sender nodes have nearly the same power consumption in a very low level

In the collect-view of the attack simulation:

- Node 1 in color green is a sink node which acts as a border router
- Nodes in color yellow are sender node which act as sensors
- Node 12 is replaced with a malicious node

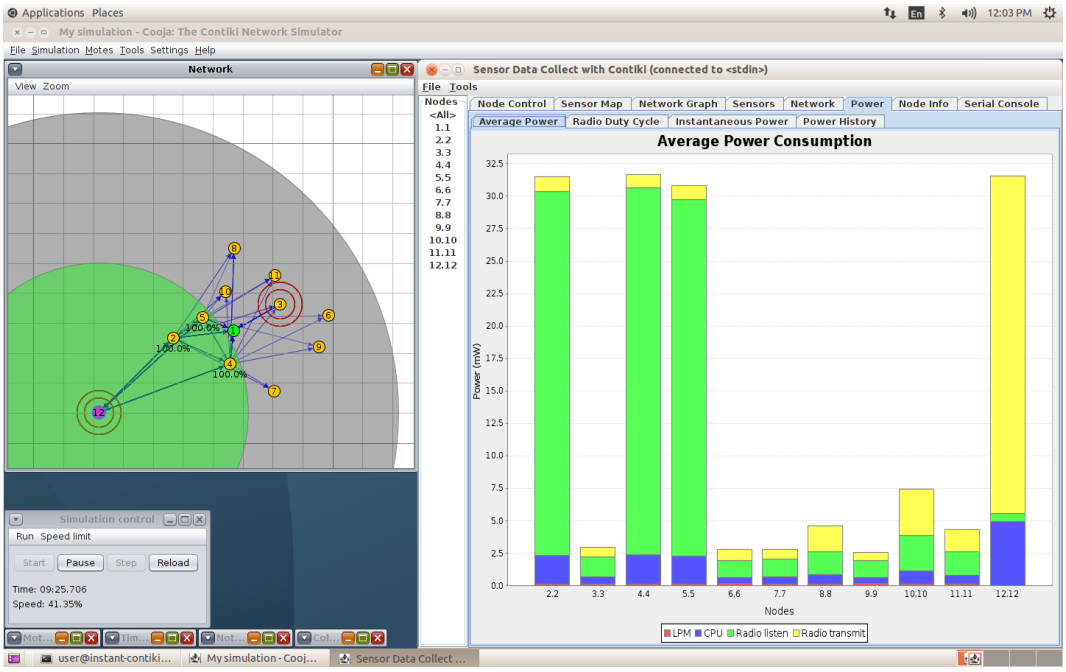


Figure 3.13: Attack simulation of flooding attack

- Node 2,4 and 5 are in the range of node 12
- Compared with other sender nodes, node 2, 4, 5 and 12 have significantly high power consumption, and other sender nodes' power consumption are also higher than before
- Among the power consumption of node 12, the Radio transmit consumes a large proportion of the power consumption, because it continuously send messages to the node 2, 4, and 5
- Among the power consumption of node 2, 4, and 5, the Radio listen consumes a large proportion of the power consumption because they continuously receive messages from the node 12

Through this kind of comparative simulation, users can easily understand various types of attacks, which is very helpful to enhance their security awareness.

### 3.2.4 System Structure Design & Implementation

As shown in figure 3.1, the system structure of IoTrain consists of a database, an interface, some function modules. The training content de-

scribed above is the data stored in the database. The following content will be divided into two parts, the IoTrain database will be introduced in the first part, and the interface and function modules will be introduced in the second part.

### Database

Due to the entire system currently has only three file types and the number of files is small. Besides, the C files involved are all located inside the Contiki OS, only the PDF format tutorial files and CSC format simulation files are stored in the database. Therefore, the database is built without using professional database management software, and all files are classified and stored in folders and subfolders of the entire project.

Figure 3.14 shows the folder tree of IoTrain. All blue blocks represent folders, and all purple blocks represent files. All the blocks named "Instruction", namely the folder named "Instruction", store the PDF tutorials. Similarly, all blocks named "Simulation" (namely the folder named "Simulation") store the CSC simulation files. Therefore, the purple block is omitted in the figure.

Since the entire project has been uploaded to the Github, the README.md file shown in the figure is the project introduction file in the Github.

The "rpl-collect folder" is a folder inside Contiki OS, and this folder is used and stored in the IoTrain project. The "rpl-collect" folder in the IoTrain project stores some modified files and some files for making security training simulations.

The "Command" folder shown in the figure stores the Python files that implement the interface and functions. These will be explained in the following section.

### Interface and Functions

The Python files in the "Command" folder that implement the interface and functions will be explained in this section.

The IoTrain interface is implemented as prompt options in the terminal window (Shown in Figure 3.15). The tutorial in Figure 3.9 is the tutorial that was automatically opened after selecting option 1 in this figure.

Functions include configuring the system environment, displaying interface and options, as well as open tutorials and partial simulation files according to users' options. The following class diagram (Figure 3.16) shows all the classes in the Command folder.

The class IniManFolder has only four string type attributes:

- _iotrain_path: The path of the rpl-collect folder in the IoTrain project.

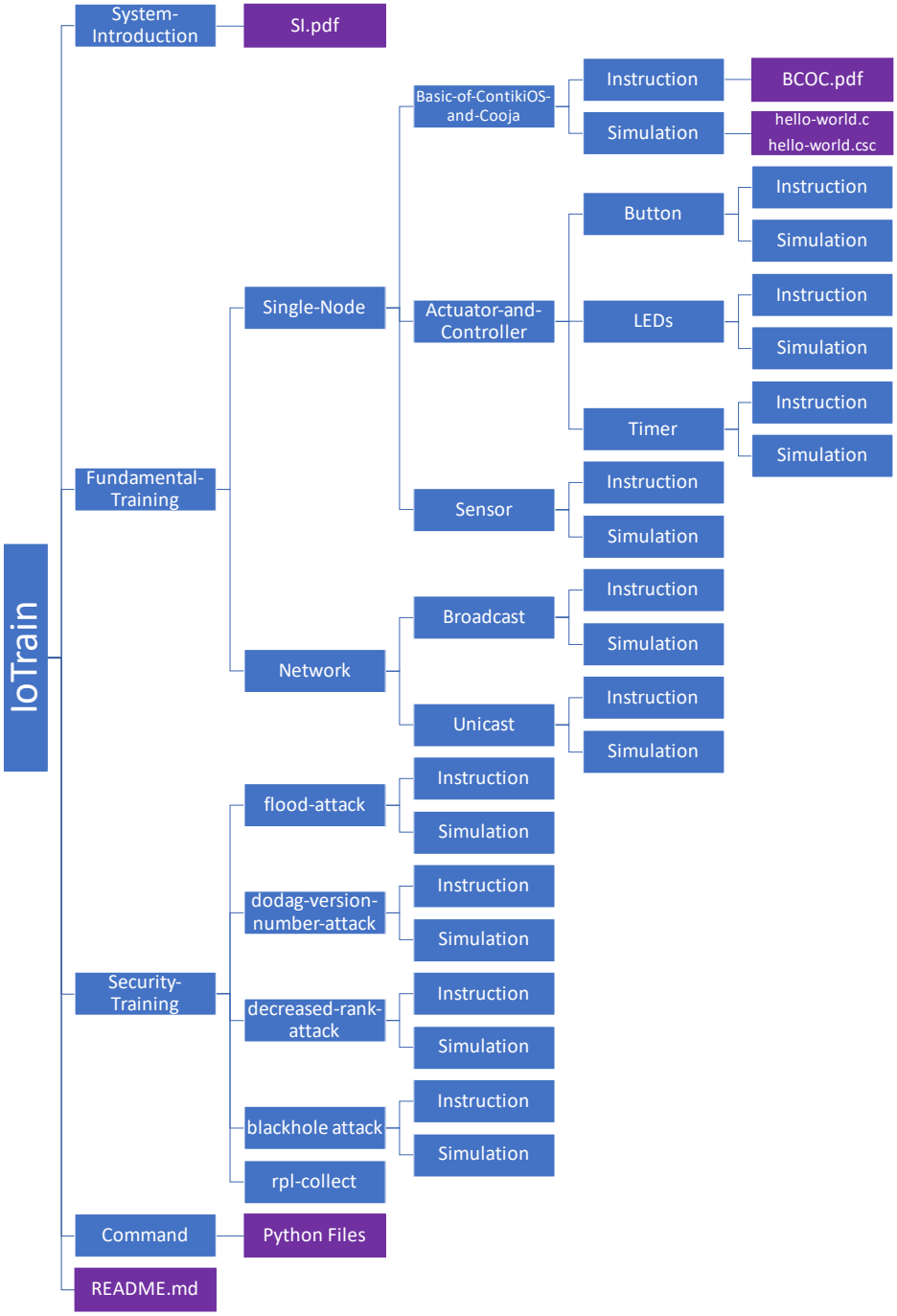


Figure 3.14: Folder tree

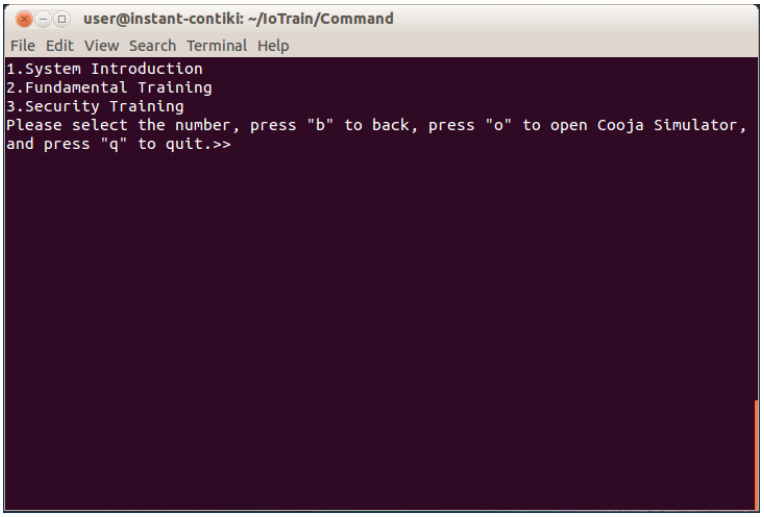


Figure 3.15: Interface

- _iotrain_backup_path: The backup path of the rpl-collect folder in the IoTrain project.
- _contiki_path: The path of the rpl-collect folder in Contiki project.
- _contiki_backup_path: The backup path of the rpl-collect folder in Contiki project.

The class ManageFolder has five attributes and three methods. The attribute _IniManFolder is the instance of the class IniManFolder. Other four attributes respectively store the values of the four attributes in class IniManFolder. Method backup_folder() is to back up the rpl-collect folder, and method move_folder() is to move the rpl-collect folder in IoTrain project to Contiki project and overwrite the rpl-collect folder with the same name in the Contiki project. Method recovery() is to restore the changes caused by the previous two methods. This method is used when exiting the system.

The class Content has only one dictionary type attribute that maps out the training content structure and stores the file names of all training content files.

The class OpenCooja has only one method that is used to open the Cooja network simulator.

The class ShowMenus is the core class of the system, whose attributes contains instances of the class Contents, the class ManageFolder, and the class OpenCooja. As shown in the Class Diagram, it has the following method:

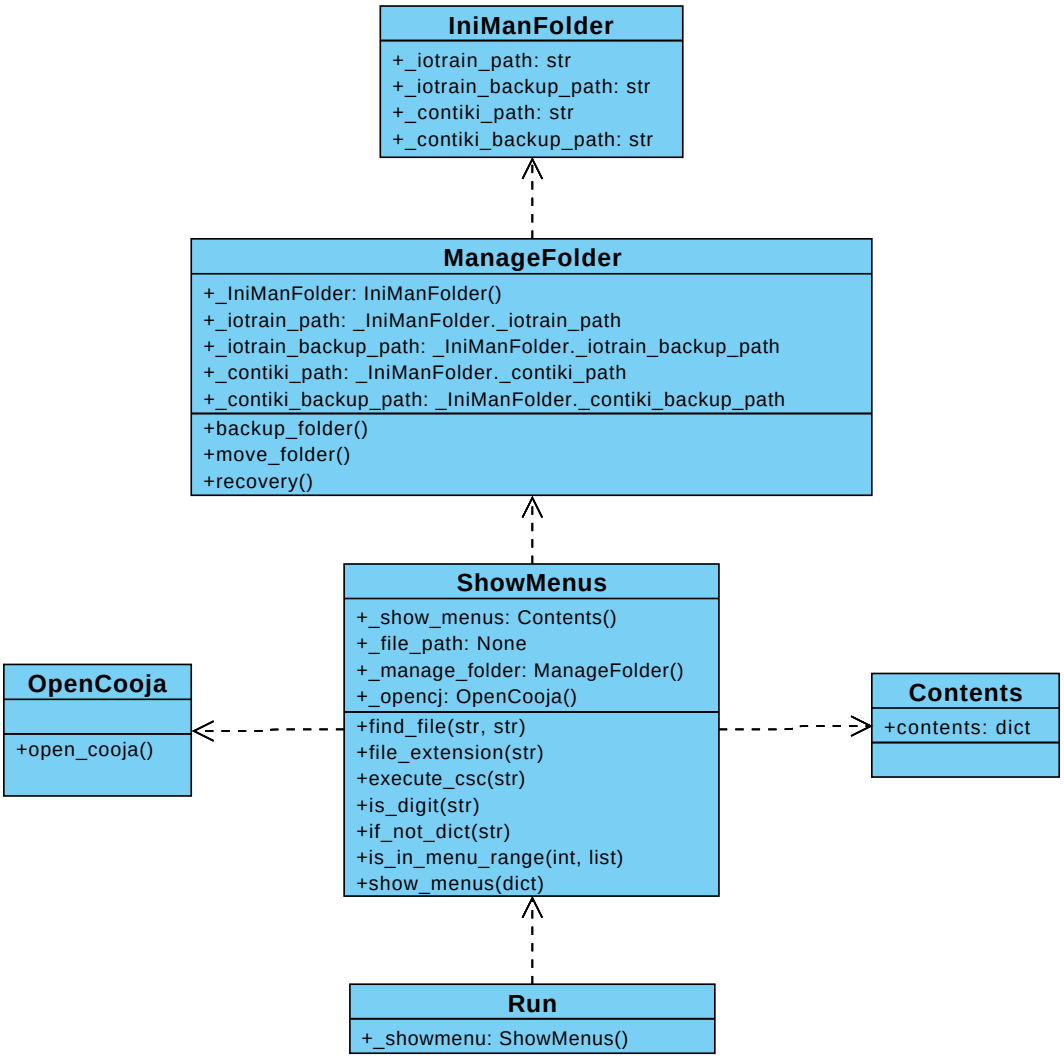


Figure 3.16: Class diagram

- find_file(str, str): Find the file and return the path according to the file name parameter.
- file_extension(str): Identify the file name passed in and return the extension of the file name.
- execute_csc(str): Open the files with the "csc" extension.
- is_digit(str): Judge whether the string content entered by the user is a number, and if so, converts the string type to an integer type.
- if_not_dict(str): Judge if the numeric option entered by the user corresponds to a dictionary type content.
- is_in_menu_range(int, list): Judge if the user's input is legal.
- show_menus(dict): Use the recursive algorithm to hierarchically output the training content, which is stored in a Python dictionary.

The class Run is the entry to the system, and its attribute "_showmenus" is an instance of class ShowMenus. Class Run is written in a Python file called run.py. There is also an instance of the class Run in this file. When run.py is executed, the class Run is instantiated first, and then the instance of class Run calls the backup_folder() method and move_folder() method to initialize the system. Then it calls show_menus() method, which will print out the system interface in the terminal window. The show_menus() method also calls various other methods to implement various functions of the system based on user input.

# Chapter 4

# System Evaluation

This chapter will focus on the evaluation of the system. The evaluation will start with three aspects: feature, performance, and achievement on system requirements.

## 4.1 Feature Evaluation

The method of feature evaluation is to compare IoTrain with other existing IoT training systems. I have found two complete and representative systems from all IoT training systems I have investigated to compare with IoTrain. These two systems are IBM Watson IoT Online Academy (IWIOA) [22] provided by IBM, and the IoT Training System (ITS) provided by 3 Rocks Technology [23].

IBM is an American multinational information technology company that produces and sells computer hardware, middleware, and software. IBM also provides hosting and consulting services in areas ranging from mainframe computers to nanotechnology. IWIOA stands for online learning systems. The training content or courses of such IoT training systems developed by commercial companies are usually tailored to their related products. Users are often potential customers who will continue to use relevant paid services for development after training. IWIOA aims to train users to develop the IoT based on the IBM Watson IoT Platform. The IBM Watson IoT Platform is a platform that helps users to get their IoT projects started.

3 Rocks Technology is an engineering training system provider. Its product lines include Electronic, Pneumatic, Hydraulic, Mechatronics, Communication Circuits training systems. ITS is a series of IoT training experimental boxes (shown in Figure 4.1 and 4.2) that integrated with a lot of hardware, such as CPU Board, Gateway (Raspberry Pi), sensors, and actuators. ITS

stands for hardware-based physical training system. Such products or systems are usually group-oriented, such as schools or training institutions, and are therefore often expensive and unaffordable for individual users.

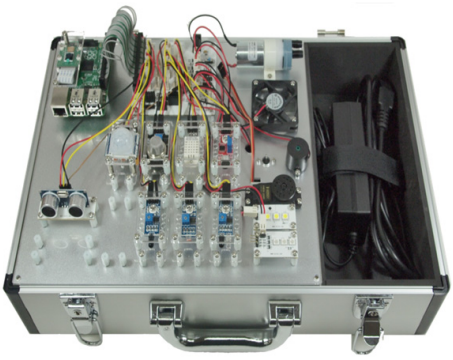

Figure 4.1: IoT experimental box

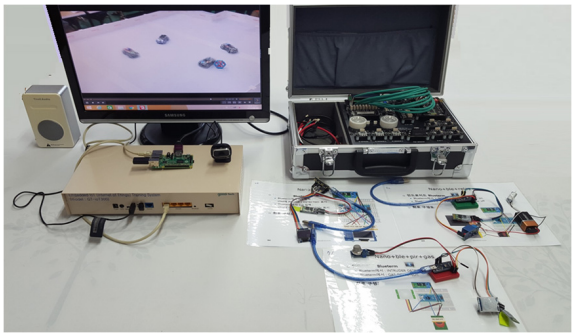

Figure 4.2: IoT application kits

The Table 4.1 shows the comparison between the three IoT training systems.

The column on the left side of the table is the comparison items, where:

- Obtain refers to how users get the system.
- Register refers to whether the user needs to register or log in.
- Internet Requirement refers to whether users need to access the Internet when using the system.
- Capital cost refers to whether the user needs to pay for the use of the system.
- Knowledge threshold is the level of knowledge a user needs to use the system.
- Content form refers to the form in which training content is presented to the user.
- Device form means the form of the device used in the system, including virtual devices and real devices.
- Device type refers to the type of device supported in the system.
- Operating system represents the operating system used in the system.
- Security training refers to whether the system training content includes security training.

- Large-scale WSNs refers to the difficulty of using this system to build large-scale wireless sensor networks.

Table 4.1: Feature comparison

| System / Feature | IoTrain | IWIOA | ITS |
|---|---|---|---|
| **How to obtain** | Free download | Online learning | Purchase equipment |
| **Need to register** | ✗ | ✓ | ✗ |
| **Internet requirement** | Offline | Online | LAN |
| **Cost** | Free | Free | High cost |
| **Target** | Anyone | Employee or business partner | Employee or student |
| **Pre acquisition knowledge** | Low | Medium | Secondary or higher education |
| **Available content** | Tutorial and simulation | Video, simulation and document | Smartphone application examples |
| **Device form** | Virtual devices | Virtual devices | Real devices |
| **Device type** | 3 kinds of sensor, 1 kinds of actuator(currently) | All devices supported by the IBM Cloud Platform | 8 kinds of sensor, 7 kinds of Actuator |
| **Operating system** | Contiki OS | All systems supported by the IBM Cloud Platform | Debian GNULinux, Arch Linux ARM, RISC OS |
| **Security training** | ✓ | ✓ | ✓ |
| **Large-scale WSNs** | Easy | General | Difficult |

As can be seen from the table, the advantages of IoTrain are low cost, low threshold, and support for large-scale WSNs. However, because IoTrain is a newly developed system and the developer level is limited, it has a disadvantage regarding content richness and device richness compared to the other two systems.

## 4.2 Performance Evaluation

The performance evaluation will focus on the time it takes to start Instant Contiki and IoTrain, open tutorials, and simulations. The test is done by turning on a timer at the start of the evaluation and stopping the timer at the end of the evaluation. Since the start timer and the stop timer are both manual, there is an artificial delay in response. To increase the accuracy, each test value is the average of three test values. Even so, the test results are still approximate values.

Table 4.2: Environment characteristics evaluation

| **Item** | **Detail** |
|---|---|
| Computer | MacBook Air (13-inch, 2017) |
| Processor | 2.2 GHz Intel Core i7 |
| Memory | 8 GB 1600 MHz DDR3 |
| Graphics | Intel HD Graphics 6000 1536 MB |
| OS Version | macOS Mojave 10.14.2 (18C54) |
| VMware | Fusion Professional Version 10.1.3 (9472307) |
| Instant Contiki | Instant Contiki 3.0 |
| Ubuntu | 14.04 LTS |
| Python | Python 2.7.6 |
| Contiki | Contiki 3.0 |

The table 4.2 lists the hardware and software parameters below. All development is done on the MacBook Air. Because the latest version of Instant Contiki uses Ubuntu 14.04 images, the IoTrain system is also based on Ubuntu 14.04.

Table 4.3: Performance evaluation

| **Operation** | **Time** |
|---|---|
| Start Instant Contiki | 26.3s |
| Start IoTrain | 0.5s |
| Open a tutorial | 0.7s |
| Open a simulation | 3.3s |
| Complete a simulation | 8s ... 20min |
| Quit IoTrain | 0.3s |
| Shutdown Instant Contiki | 4.9s |

The table 4.3 shows evaluation results.

Start/Shutdown Instant Contiki is actually launching Ubuntu 14.04 image in VMware. The time required depends on the computer hardware and has nothing to do with the IoTrain system.

Complete a simulation is essentially a call to the Cooja simulator, and the time required is highly dependent on the content of the simulation. The more motes that are added to the simulation, the longer it takes to complete the simulation, especially for attack simulation. Attack simulations can consume a lot of computer resources and can therefore be very slow.

## 4.3 Requirement Evaluation

The evaluation in this section is based on the system requirements in 3.1. First of all, I will give a comparison evaluation table (Table 4.4) of the three systems. Then I will just list explanations for every evaluation result of the IoTrain system.

Table 4.4: Requirement evaluation comparison

| **System** / **Requirement** | IoTrain | IWIOA | ITS |
|---|---|---|---|
| Open-source | ✓ | | ✓ |
| Parallel | | ✓ | |
| For different level users | ✓ | | |
| Content-rich | | ✓ | ✓ |
| Low cost | ✓ | | |
| Easy to manage | ✓ | ✓ | |

- **Open-source**

  After the system is completed, the source code will be uploaded to the Github repository for everyone to use and develop.

- **Parallel**

  This requirement has not been met. Currently, the system only supports single user downloads and then runs independently on the virtual machine software of the PC. But if multiple such images can be quickly generated on one or more servers, numerous users can simultaneously access independently, and thereby the IoTrain can meet the parallel requirements.

- **For different level users**

  This requirement have been basically met. However, because the current training content is still too small, the users covered is still very limited.

- **Content-rich**

  Content is at the heart of all IoT training systems. Because the developer level is limited and the system development time is limited, the current system content has not reached a rich level. But the system's database is simple and easy to manage, so if the developer and time are sufficient, the lack of system content will soon be improved.

- **Low cost**

  Thanks to open source development tools and the use of simulators to simulate hardware, there is only a time cost for development.

- **Easy to manage**

  The current database design is very simple, and the content is not much. The management of the training content is essentially a modification of the Python dictionary content, so it is still easy to manage. But as the training content increases, management will become difficult. It will become necessary to use the database software by then.

# Chapter 5

# Conclusion

The idea of this research came from reading some papers about cybersecurity training. Besides, I was very interested in the IoT at that time, so I started to survey the IoT security training. However, I found that the IoT security status is not optimistic, because there are few studies on the IoT security training. This situation gave me the motivation to start this research.

The research process was not smooth, and there were some failed attempts in the early stage. But under the guidance of my supervisor, the research was soon on the right track.

In the whole research, the most important contribution is the proposal of the training content structure. It is the blueprint of IoTrain's development. Besides, it not only has a high reference value for the future development of the IoTrain system but also for the development of other IoT training systems.

The development of the IoTrain system based on the training content structure is the second contribution of this research. After proceeding the various stages, the system has reached the prototype stage, where the system's training content structure, functions, and interfaces have been implemented.

Using simulation in IoT training is indeed a useful attempt. The use of simulators not only reduces the cost of development, but also increases the training content forms, while also making training more interesting and effective.

As can be seen from the various previous evaluations, there are many advantages of the IoTrain system.

From the perspective of feature evaluation, the most distinct advantage is low cost and easy to use. It can be freely obtained and easily used by anyone. If this system can be promoted and widely used, then the popularity of IoT

security knowledge will be of great benefit.

In terms of performance evaluation, IoTrain is a very lightweight system, and the entire system is currently only about 80 megabytes. The IoTrain system runs on a Linux system known for stability. Whether it is started or performs various functions, tasks can be done in few seconds.

As for the requirement evaluation, the best of the three systems to meet the system requirements is the IoTrain system. Among the six requirements, IoTrain met four of them, with the highest completion rate of 66%, and the two unmet are not impossible to achieve.

However, IoTrain is not perfect either. To make IoTrain a qualified training system for IoT, there is still much work to be done in the future.

The first is that there is too little training content. On the one hand, because the developer's level is too limited to fully utilize the development tools, the development is still in an early stage. On the other hand, the development tools are limited to the RPL based IoT, so the IoTrain can only be considered as a branch training system of IoT.

The second is parallel mentioned in system requirements. Because the Contiki OS, the Cooja network simulator and the IoTrain system are all stored in an Ubuntu virtual machine image, if multiple such images can be quickly generated on one or more servers, numerous users can simultaneously access independently, and thereby the IoTrain can meet the parallel requirements.

The third is automation, including automation of system content generation and automation of system management. The former means that the automatically generate corresponding training content according to the keyword input by the user, without the developer making and storing it in advance. The latter refers to automatically managing the training content of each user.

Combined with the advantages and disadvantages of the IoTrain system, it can be concluded that the development of the IoTrain is a useful attempt to develop the IoT training system. The development and implementation process of the system have high reference value, and the system also has a considerable application value. With further development in the future, I believe that IoTrain will become a more comprehensive and powerful IoT training system.

# Bibliography


[1] Kanellos, M., (2016). 152,000 Smart Devices Every Minute In 2025: IDC Outlines The Future of Smart Things. Retrieved on January 6,2019 from https://www.forbes.com/sites/michaelkanellos/2016/03/03/152000-smart-devices-every-minute-in-2025-idc-outlines-the-future-of-smart-things/\#66501aac4b63.

[2] Williams, C., (2016). Today the web was broken by countless hacked devices. Retrieved on January 6, 2019 from https://www.theregister.co.uk/2016/10/21/dyn_dns_ddos_explained/.

[3] Internet of Things Research Study. (2014). Retrieved on January 7, 2019 from http://d-russia.ru/wp-content/uploads/2015/10/4AA5-4759ENW.pdf.

[4] Pathak, P., Vyas, N., Joshi, S., (2017). Security Challenges for Communications on IOT & Big Data., in *International Journal 8*, 431–436.

[5] Security Awareness in the Age of Internet of Things. (2016). Retrieved on January 7, 2019 from https://download.bitdefender.com/resources/files/News/CaseStudies/study/136/Bitdefender-Whitepaper-IoTSecurity-A4-en-EN-web.pdf.

[6] Ramirez, E., (2015). Privacy and the IoT: Navigating Policy Issues. Retrieved on January 7, 2019 from https://www.ftc.gov/system/files/documents/public_statements/617191/150106cesspeech.pdf.

[7] Xu, X., (2013). Study on security problems and key technologies of the internet of things, in *Proceedings - 2013 International Conference on Computational and Information Sciences, ICCIS 2013* (pp. 407–410).

[8] Contiki: The Open Source OS for the Internet of Things. Retrieved on January 19, 2019 from http://www.contiki-os.org/.

[9] Get Started with Contiki. Retrieved on January 19, 2019 from http://www.contiki-os.org/start.html.

[10] Cuong, P. D., (2017). On Automatic Cyber Range Instantiation for Facilitating Security Training. Retrieved on January 10, 2019 from https://dspace.jaist.ac.jp/dspace/bitstream/10119/14161/3/paper.pdf.

[11] Internet of things. Retrieved on January 11, 2019 from https://en.wikipedia.org/wiki/Internet_of_things#cite_note-4.

[12] ITU-T Recommendation Y.2060 (2012). Overview of the Internet of things.

[13] Rosencrance L., Shea S., and Wigmore, I., (2018). internet of things (IoT). Retrieved on January 11, 2019 from https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT.

[14] K. Zhao and L. Ge. A survey on the internet of things security, in *Computational Intelligence and Security (CIS), 2013 9th International Conference*, on pp. 663–667, IEEE, 2013.

[15] M. Wu, T. J. Lu, F. Y. Ling, J. Sun, and H. Y. Du. Research on the architecture of Internet of Things, in *Proc. 3rd ICACTE, 2010*, pp. V5-484–V5-487.

[16] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials, 17(4), 2347–2376.*

[17] Scully P., (2018). The Top 10 IoT Segments in 2018 – based on 1,600 real IoT projects. Retrieved on January 13, 2019 from https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/.

[18] Yogita Pundir, M., Sharma, M. N., & Singh, Y., (2016). Internet of Things (IoT): Challenges and Future Directions. *International Journal of Advanced Research in Computer and Communication Engineering, 5(3)*, 960–964.

[19] I.F. Akyildiz and W. Su and Y. Sankarasubramaniam & E. Cayirci, (2002). Wireless sensor networks: a survey. *Computer Networks, 38(4)*, 393 - 422.

[20] Carl A. Sunshine. Source routing in computer networks, p. 29.

[21] Mayzaud A., Badonnel R., Chrisment I. A Taxonomy of Attacks in RPL-based Internet of Things. *International Journal of Network Security, IJNS, 2016, 18 (3)*, pp.459 - 473.

[22] Welcome to the IBM Watson IoT Online Academy. Retrieved on January 27, 2019 from `https://www.iot-academy.info/index.php/en-us/`.

[23] Homepage of 3 Rocks Technology. Retrieved on January 27, 2019 from `https://www.3rockstech.com/`.

[24] R. Beuran, C. Pham, D. Tang, K. Chinen, Y. Tan, Y. Shinoda, CyTrONE: An Integrated Cybersecurity Training Framework. *International Conference on Infor- mation Systems Security and Privacy (ICISSP 2017)*, Porto, Portugal, February 19-21, 2017.