

## **COPYRIGHT WARNING**

This paper is protected by copyright. You are advised to print or download **ONE COPY** of this paper for your own private reference, study and research purposes. You are prohibited having acts infringing upon copyright as stipulated in Laws and Regulations of Intellectual Property, including, but not limited to, appropriating, impersonating, publishing, distributing, modifying, altering, mutilating, distorting, reproducing, duplicating, displaying, communicating, disseminating, making derivative work, commercializing and converting to other forms the paper and/or any part of the paper. The acts could be done in actual life and/or via communication networks and by digital means without permission of copyright holders.

The users shall acknowledge and strictly respect to the copyright. The recitation must be reasonable and properly. If the users do not agree to all of these terms, do not use this paper. The users shall be responsible for legal issues if they make any copyright infringements. Failure to comply with this warning may expose you to:

- Disciplinary action by the Vietnamese-German University.
- Legal action for copyright infringement.
- Heavy legal penalties and consequences shall be applied by the competent authorities.

The Vietnamese-German University and the authors reserve all their intellectual property rights.





RUHR-UNIVERSITÄT BOCHUM

**MechEng**  
Mechanical Engineering



Vietnamese-German University

# Design of a Smart Waste Management System for a University Campus

Bachelor Thesis  
BINH DUONG 2023



Vietnamese-German University

**Submitted by:** Nguyen Minh Tri

**RUB Student ID:** 18206061

**VGU Student ID:** 11982

**Supervisor:** Dr. Nguyen Hong Vi

**Co-supervisor:** MSc. Chau Khac Bao Chuong

# Design of a Smart Waste Management System for a University Campus

A Thesis Presented

By

Nguyen Minh Tri

Submitted to the department of Mechanical Engineering of the  
RUHR-UNIVERSITÄT BOCHUM and VIETNAMESE-GERMAN UNIVERSITY  
in partial fulfillment  
Of the requirement for the degree of  
BACHELOR IN MECHANICAL ENGINEERING

September 2023

Major: Mechanical Engineering

# Affirmation in lieu of oath

Nguyen Minh Tri

Matriculation Number: 11982

Title of Thesis: Design of a Smart Waste Management System for a University Campus

I hereby declare in lieu of oath that I have produced the aforementioned thesis independently and without using any other means except the aids listed. Any thoughts directly or indirectly taken from somebody else's sources are made discernible as such.

To date, the thesis has not been submitted to any other board of examiners in the same or a similar format and has not been published yet.



Binh Duong, 05/09/2023

A handwritten signature in black ink, appearing to be 'Nguyen Minh Tri', is written over a horizontal line.

Nguyen Minh Tri

# Abstract

With the growth of the current world population and the increase in consumerism, waste management has become a major global concern, especially in the wake of the Covid-19 pandemic. Densely populated facilities such as dormitories, apartment buildings, or university campuses can produce astoundingly large amounts of unsorted waste, which eventually end up in large landfills and have a negative impact on the environment. A smart waste management system installed in these locations would attenuate the negative impacts mentioned, thus creating a safer and more hygienic environment. This thesis aims to design a system that improves waste management by regularly monitoring waste levels and generating optimized waste collection routes based on waste level data. The proposed system consists of multiple devices that monitor trash bins on a university campus and then send data to a server using LoRa technology. The data collected from the bins are then transferred to the Internet for viewing and analysis to generate a suitable waste handling path. The system also utilizes a specifically designed trash bin optimized for placement in populated areas, regarding ease of use and maintenance, and features a notification panel that reminds users to discard trash appropriately.

*Keywords:* waste management system, LoRa, monitor waste levels

# List of Figures

Figure 1 System Overview.....	15
Figure 2 NodeMCU ESP8266 .....	18
Figure 3 Heltec WiFi LoRa 32.....	19
Figure 4 LoRa Ra-02 SX1278 .....	19
Figure 5 Ultrasonic sensor IOE-SR05 .....	20
Figure 6 Humidity and Temperature Sensor DHT21 .....	22
Figure 7 Voltage Regulator LM2596.....	23
Figure 8 18650 Li-ion batteries.....	23
Figure 9 Arduino Nano .....	24
Figure 10 Nextion 7-inch display module .....	25
Figure 11 Rd-03 mmWave Human Presence Sensor.....	25
Figure 12 Data Collection Flow Chart.....	30
Figure 13 LoRa Server Flow Chart.....	31
Figure 14 Shortest Route generation Flow Chart .....	33
Figure 15 Trash container User Notification Flow Chart .....	34
Figure 16 The assembly of the trash container .....	35
Figure 17 The upper part of the trash container.....	36
Figure 18 The bin of the trash container .....	36
Figure 19 The dock of the trash container .....	36
Figure 20 Drawing of the upper part of the trash container.....	37
Figure 21 Drawing of the bin of the trash container .....	37
Figure 22 Drawing of the dock of the trash container .....	37

Figure 23 Assembly of a monitoring unit.....	38
Figure 24 The Heltec Wi-Fi LoRa enclosure, with smalling mounting points.....	39
Figure 25 Drawing for the Heltec Wi-Fi LoRa enclosure .....	39
Figure 26 The NodeMCU enclosure, with larger mounting points .....	40
Figure 27 Drawing for the NodeMCU enclosure .....	40
Figure 28 View of the top cover for the enclosures.....	41
Figure 29 Drawing of Top cover .....	41
Figure 30 A preview of components installed in a monitoring unit .....	42
Figure 31 The experimental assembly of the monitoring unit.....	42
Figure 32 An example Traveling Salesman Diagram.....	43
Figure 33 Example 4 nodes.....	45
Figure 34 Branch-and-bound example.....	46
Figure 35 Nearest-neighbor method example.....	47
Figure 36 Space-Filling Curve Example.....	48
Figure 37 Ant colony optimization example .....	50
Figure 38 TSP-Solver 6 nodes .....	51
Figure 39 The TSP-Solver running.....	51
Figure 40 Result from the TSP-Solver.....	52
Figure 41 TSP-Solver 12 nodes .....	52
Figure 42 Path Generation Flow Chart .....	54
Figure 43 The enclosure before installation.....	56
Figure 44 Heltec Wi-Fi LoRa enclosure fully installed.....	57
Figure 45 NodeMCU enclosure fully installed.....	57
Figure 46 Heltec Wi-Fi LoRa board as the server .....	58

Figure 47 Experiment Placement..... 58

Figure 48 Experiment monitoring result..... 63

Figure 49 Measured current ..... 64





# List of Abbreviations

Abbreviation/Acronym	Is shortened for
RH	Relative Humidity
IoT	Internet of Things
VGU	Vietnamese – German University
TSP	Traveling Salesman Problem
W	Watt
A	Amp
V	Voltage
kHz	Kilohertz
MHz	Megahertz
$\mu$ s	Microseconds
mm	Milimeter
g	Gram
ACO	Ant Colony Optimization
mAh	Miliamphour
mA	Milliamp
IDE	Integrated Development Environment

# List of Tables

Table 1 Bill of Materials for NodeMCU monitoring unit .....	26
Table 2 Bill of Materials for Heltec LoRa ESP32 monitoring unit .....	27
Table 3 Bill of Materials for server unit .....	27
Table 4 Bill of Materials for components in the trash container .....	28
Table 5 Sensor connections .....	55
Table 6 Connections of LoRa module to microcontroller .....	56
Table 7 Bill of Materials for experimental set up .....	65



# Contents

Affirmation in lieu of oath .....	2
Abstract .....	3
List of Figures .....	4
List of Abbreviations .....	7
List of Tables .....	8
1. Introduction .....	11
2. Literature Review .....	13
3. Methodology.....	15
3.1 Principle Theory.....	16
3.1.1 System Architecture.....	16
3.1.2 System Components.....	17
3.1.3 Bill of Materials:.....	26
3.1.4 Processes .....	29
3.2 Technical Design.....	35
3.2.1 Trash container design .....	35
3.2.2 Design of the monitoring unit enclosure.....	38
3.3 Algorithms.....	43
3.3.1 Definition of the problem.....	43
3.3.2 Solutions to the problem .....	43
3.3.3 The brute force method.....	43
3.3.4 Branch and bound method .....	45
3.3.5 Nearest-Neighbor method.....	47
3.3.6 Space-Filling Curve Method.....	47

3.3.7	Ant Colony Optimization.....	48
3.3.8	Trial runs.....	50
3.3.9	Conclusion .....	53
3.4	Experimental Setup .....	55
4.	Results .....	63
4.1	Data transfer .....	63
4.2	Power Consumption.....	64
4.3	Bill of Materials for the experiment.....	65
5.	Discussion and Conclusions .....	66
6.	References .....	68



# 1. Introduction

According to the World Bank [1], cities around the world created 2.01 billion tons of solid waste in a single day in 2016, resulting in a footprint of 740 g per person. Due to rapid population expansion and urbanization, annual waste generation is estimated to increase by 70 % from 2016 to 3.40 billion tons in 2050, according to a publication by World Bank [2].

At such a rate, one of the many challenges a well-attended university faces is the management of the waste generated by its students and faculty. Waste is defined as discarded usable items or unusable materials in their current form. Recyclable waste is best divided into twelve categories [3]. Waste comes in various forms, such as solid, liquid, or even gaseous, and every form of waste requires different classifications, disposal techniques, and management. Therefore, improper practices are detrimental to the environment and can cause further problems, such as mixing hazardous wastes or creating harmful by-products.

The existing waste management system present at the Vietnamese-German University, where trash is collected daily from bins around the campus, is insufficient to categorize recyclable waste and separating organic waste for other uses. The requirements for the smart management system this thesis seeks to design are as follows:

1. Ease of categorization of trash for students and faculty, even without prior environmental knowledge of trash segregation.
2. Efficient collection and transfer of data across a large university campus.
3. Optimization of waste collection routes based on sensor data.

The resulting smart waste management system proposed in this thesis would feature an array of trash bins throughout the campus, from dormitories to hallways and cafeterias, each bin categorizing trash into two types: organic and inorganic. Dividing waste into these two types eliminates the confusion caused by multiple options, resolving the first requirement. The benefit of this design is that the separation of organic and inorganic waste makes the subsequent classification of inorganic waste for recyclables much more hygienic and convenient. An additional feature of the custom design is the detection of nearby users to send notifications reminding said users to dispose of trash in the correct bin. Each container is equipped with a

module that contains a trash level sensor, a humidity and temperature sensor, and a circuit to collect and transmit data over long distances using LoRa technology.

The second requirement dictates a transmission technology capable of covering a large university campus, including its dormitories. Thus, LoRa is the optimal choice, due to its long range, low power consumption, and stability. Transmitting data over a large area with LoRa is more stable than other methods such as over Wi-Fi, where coverage can be limited or even non-existent. The module can be configured to send waste bin fill level data every hour for three to five minutes, after which the device can go to sleep until the next transmission phase. At night, when activity is low, the module can save much more energy by being on standby until morning. With such low power consumption, a module can transmit data for months on end before needing a battery replacement. Therefore, LoRa is the key to efficiently transmitting data across a large university campus.

Finally, the collected data is transmitted to a management center and analyzed for optimal trash collection schedules. Since the placement of trash containers and the building layout of a university campus is static, a map can be drawn with nodes representing the trash bin locations and the status of the said nodes is binary: collection needed or not full. With the mapping of full nodes, multiple algorithms can be applied to generate several waste collection routes, which are then compared to determine the most efficient path. Since all full bins must be emptied, the collection route becomes a Traveling Salesman Problem (TSP) [4], whose complexity increases with more nodes in the system. However, there are many algorithms available to address this problem to generate the shortest path to collect trash.

## 2. Literature Review

Waste management involves the handling, transportation, processing and proper disposal of waste materials [5]. In the case where waste management is not handled properly, major issues can arise and pose risks to public health as well as to the environment.

In practice, conventional waste management systems face inefficiencies in the collection and disposal of trash due to manual labor and outdated technologies, leading to pollution and environmental harm. To resolve these problems, a smart waste management system must implement newer technologies to reduce valuable resources spent on redundant and manual tasks. The Internet of Things (IoT) offers a wide array of possibilities to achieve a smart system, which includes sensors, data analysis and algorithms to help monitor the status of important statistics and generate strategies to optimize waste collection. However, the components of a smart waste management system can vary depending on the actual local requirements, because different facilities or regions can generate vastly different volumes and categories of trash.

Nevertheless, these systems should have similarities, which include, but not limited to:

- Smart trash containers that can be equipped with units that monitor the fill level of the bins. These units can report the data back to a server for further processing, for example alerting when bins are full and need to be emptied. A common feature of these unit is a long-lasting battery life and long range communication.
- Data analytics: a smart waste management can perform data analysis on the data collected to identify trends and patterns, thus giving insights that could improve efficiency and even enhance waste disposal [6].
- Tracking the movement of waste from its collection to its disposal, which require the smart system to install sensors and tags [7]. After many trips are performed, the collected data can be used to optimize subsequent collection routes, thus reducing costs in terms of fuel and time spent on waste collection.

Despite many smart waste management systems having similar structures and components, some can differ greatly from others depending on where the system is implemented, or the scale at which it is deployed.

A common theme for smart waste management systems implemented in foreign countries would be to segregate wastes diversely and precisely. Organic trash would be separated into wood, compost, paper, etc. while inorganic trash would be sorted into metal, plastic, glass, etc [8]. This is efficient at the user level for larger developed countries where the recycling system is advanced and performs better as the trash input of recycling facilities come organized. The same is not true for the situation in Vietnam, where environmental protection knowledge is not yet widespread and requiring people to categorize trash at the deepest level quickly becomes confusing and suffocating. Furthermore, a complicated waste sorting system boasts high manufacturing expenses and physical bulk, often requiring camera, multiple sensors and even motors to function [7].

On the other end, instead of focusing on hardware, some research opt for the approach of using Machine Learning (ML) to categorize waste and cut down on expenses on electronic components. However, the goal is still the same: segregate waste at a deep level. While all the above methods help to alleviate the problem presented with requiring the users to accurately categorize waste, this thesis argues that a smart waste management system that touches the user end should make the disposal of waste as smooth as possible, while the resources for sorting waste should be spent elsewhere, in particular the recycling facility. Such system should monitor only the important data that affects the performance of the trash containers, and at all times be able to generate a trash collection route, after which trash is processed at a more optimized facility suitable for further segregation and recycling). Therefore, the methodology below discusses a simple smart waste management system by focusing on three key aspects of importance:

- The design of the waste container that the users interact with.
- The monitoring and transfer of data for processing of the units installed in the containers.
- The generation of an optimal waste collection route.



### 3. Methodology

The focus of this thesis is to develop a system that achieves three goals: helping users segregate trash into organic and inorganic wastes, efficiently transmitting data over a large area, and generating an optimal route for trash collection. It is proposed by carefully understanding how users handle trash. Most students at a university are neither equipped nor expected to have enough knowledge to separate trash into their respective categories. Therefore, a trash container design that solely offers the binary choice between organic and inorganic trash compartments should in theory eliminate any confusion and complicated decision making involving precisely categorizing the trash; the subsequent trash segregation should be performed at a specific facility, not weighed on the shoulders of young students. Second, taking into consideration the large area of a university campus and its enclosed dormitories, alongside the impracticality to have Wi-Fi coverage campus wide, LoRa technology arises to be the most suitable form of data transmission, featuring long range and low power consumption. Finally, to complete the smart management system, a variety of algorithms can be applied to the collected data to generate an optimal waste collection route, by injecting the algorithm through a Traveling Salesman Problem. With increased nodes, however, the complexity of the system rises, and other algorithms discussed below can be used to address the problem. The whole system can be briefly summarized by the diagram below:

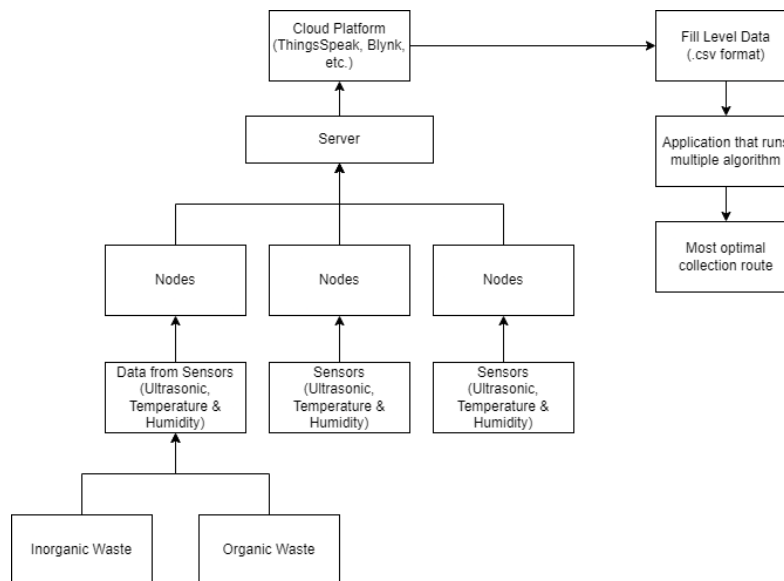


Figure 1 System Overview

## 3.1 Principle Theory

### 3.1.1 System Architecture

The system consists of several trash containers specifically designed to help users properly dispose of waste, by having sensors to detect their presence and reminding them to choose the correct bin to place the trash in. The containers are separated into two bins, organic and inorganic. By dividing waste into these simple categories, users are not required to have in-depth knowledge of waste segregation, and thus the decision to dispose of trash properly is less complicated and more convenient. The bins are equipped with ultrasonic sensors that detect any object in proximity that is present for more than 5 seconds and then display a notification that reminds the users to choose the correct bin for waste disposal. The design of the trash can should also include infographics on the cover to help users determine to which categories their waste belongs to.

Each container is equipped with a unit to monitor the current level of waste, as well as temperature and humidity, to detect anomalies, such as fire hazards or decomposing wastes. Ultrasonic and humidity-temperature sensors are used here to collect information. Each unit periodically sends data to a server using LoRa, a long-range communication protocol with low power consumption and independent of any Wi-Fi coverage. LoRa is compatible with most microcontrollers operating at the 3.3V logic level and can be powered from these microcontrollers without needing a separate power source.

The server is located in a position where Wi-Fi is available at all times to transfer the data received from each trash bin to the Internet, onto an IoT platform such as ThingsSpeak, Blynk, etc. for direct monitoring. The uploaded data can then be exported to a csv file to plug into a specific application made to handle the data and generate several short collection routes. Routes are then compared to determine the most optimal route. Several notable algorithms to solve for the shortest path through all nodes, also known as the Traveling Salesman Problem (TSP), are the Brute Force search, Branch and Bound, Nearest-Neighbor, Ant Colony Optimization, and Space Filling Curves.

### 3.1.2 System Components

The data monitoring system is a composition of the following hardware:

- **Trash container nodes:**
  - Microcontroller: NodeMCU ESP8266 or Heltec LoRa ESP32
  - LoRa module: LoRa Ra-02 SX1278 433Mhz/Built-in LoRa of the Heltec board.
  - Ultrasonic Sensor: IOE-SR05
  - Humidity and Temperature Sensor: DHT21.
  - Voltage regulator 8V to 5V.
  - Power Source 8V.
- **Server:**
  - Microcontroller: Heltec LoRa ESP32
  - LoRa module: Built-in
  - Wi-Fi Module: Built-in
  - Voltage regulator 8V to 5V.
  - Power Source 8V.
- **Trash bin enclosure:**
  - Microcontroller: Arduino Nano.
  - 7-inch Nextion display NX8048T070
  - Human Detection Sensor: mmWave Rd-03
  - Power source.

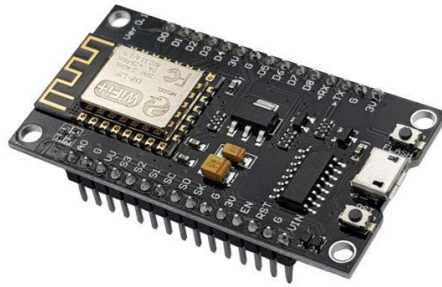


#### **Microcontrollers:**

Choosing the right microcontroller for the monitoring units require inspection of all other electronics, especially their logic levels, meaning the voltage at which the electronics send their signals. This voltage differs from operating voltage, which is the voltage of the power supply of the system. In the case of all the above components, the logic level of most sensors are from 3.3V to 5V, however, the LoRa module only works at 3.3V. Therefore, the microcontroller of choice must also operate at the 3.3V logic level to communicate properly with the LoRa module selected. Most Arduino development boards work at 5V, thus incompatible with LoRa. On the other hand, the NodeMCU ESP8266 CH340 does use 3.3V logic level, with additional advantages such as Wi-Fi, inexpensive pricepoint and wide availability [9]. It is the microcontroller of choice for the system proposed in this thesis. However, another board with

similar capabilities and built-in LoRa is also available in the form of Heltec LoRa ESP32, with LoRa communication and Wi-Fi. It is also more compact, but at the cost of an increased price point. This thesis explores both options, as well as the interaction of the 2 boards together.

- NodeMCU ESP8266:



*Figure 2 NodeMCU ESP8266*

As stated above, the NodeMCU ESP8266 CH340 microcontroller has several features useful for the system proposed in this thesis. Said features are:

- Compatibility with LoRa: the NodeMCU board operates at 3.3V logic level, the same logic level as the LoRa module of choice for this system.
- Low cost: NodeMCU ESP8266 CH340 is a very affordable development board, while being consistent and durable enough, making it suitable for commercial use.
- Open-source: NodeMCU ESP8266 CH340 is open-source, which means that the design and firmware are freely available for anyone to use and modify.
- Ease of programming: the NodeMCU board can be programmed using Arduino IDE, which is a popular and well documented programming environment.

- Heltec Wi-Fi LoRa 32 v2:

The Heltec ESP32 LoRa board is a development board that combines the ESP32 microcontroller, which has Wi-Fi, and the SX1276 LoRa transceiver chip [10]. Because LoRa and Wi-Fi is integrated into the board, it is suitable for use as a server to receiver LoRa signals, or for reducing the size of the monitoring unit that it is installed in. This board is also open-source and

programmable with Arduino IDE, convenient for development. The price point of this board is steeper than that of the NodeMCU board above but can still be considered suitable for the system proposed, due to the components integrated.

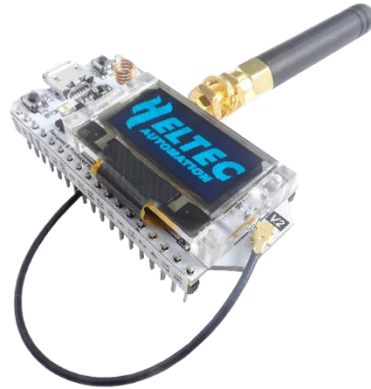


Figure 3 Heltec WiFi LoRa 32

### LoRa module:

LoRa (Long Range) is a wireless communication technology that is designed to transmit data over long distances with low power consumption. The advantages of using LoRa are:

- Long range: LoRa can be used to transmit data over distances of up to 15 km in open areas.
- Low power consumption: LoRa devices can operate for months on a single battery.
- Robust: LoRa is very resistant to noise and interference.
- Secure: LoRa uses encryption to protect data from unauthorized access.

### LoRa Ra-02 SX1278:



Figure 4 LoRa Ra-02 SX1278

Many LoRa modules are available on the market with all the aforementioned characteristics and are compatible with a wide variety of microcontrollers. However, the model Ra-02 SX1278 is the optimal choice for use with the NodeMCU ESP8266 board due to their similarities in size and their operational logic level at 3.3V [11]. Given that there must be one LoRa module in every monitoring unit, the affordable price point of the LoRa Ra-02 module is also an advantage, while still maintaining a robust design.

### Sensors:

- Ultrasonic Sensor IOE-SR05:

The IOE-SR05 ultrasonic distance sensor works on the principle of SONAR (Sound Navigation and Ranging). The sensor emits a high-frequency sound wave (ultrasound) and then listens for the echo. The time it takes for the echo to return is used from calculate the distance to the object. The IOE-SR05 has two transducers: a transmitter and a receiver. The transmitter emits an ultrasound pulse at a frequency of 40 kHz. The receiver listens for the echo and produces an output pulse whose width is proportional to the distance of the object in front [12].



*Figure 5 Ultrasonic sensor IOE-SR05*

The IOE-SR05 can measure distances between 0 mm and 2000 mm with an accuracy of 1 mm.

The basis for how the IOE-SR05 works is as follows:

1. The trigger pin is set HIGH for 10  $\mu$ s.
2. The transmitter emits an ultrasound pulse at a frequency of 40 kHz.
3. The receiver listens for the echo and produces an output pulse whose width is proportional to the distance of the object in front.
4. The width of the output pulse is measured and converted to a distance.

Sound travels at approximately 340 meters per second. This corresponds to about 29.412μs (microseconds) per centimeter. To measure the distance the sound has traveled, we use the formula:

$$Distance = \frac{1}{2}(time \times speed\ of\ sound)$$

Because sound must travel back and forth for the sensor to register, the distance is divided in half. First, sound travels away from the sensor, and then it bounces off a surface and returns. The easiest way to read the distance in centimeters is to use the formula:

$$centimeters = \frac{\frac{Microseconds}{2}}{29}$$

For example, if it takes 2000μs (microseconds) for the ultrasonic sound to bounce back, then the distance is ( $\frac{2000}{2} / 29$ ) centimeters or about 34 centimeters.

The data returned by this sensor when implemented in the smart waste management system is the distance between the sensor and the trash level. Therefore, the formula to calculate the fill level of a trash container installed with this monitoring unit is as follows:

$$fill\ level = \frac{h + l - d}{h} \times 100\ (percent)$$

Where:

h is the height of the trash container, where at the height h the trash level is 100 percent.

l is the distance between the monitoring unit and the highest trash level.

d is the measured distance from the monitoring unit to the current trash level.

For example: assume a trash container with height 80cm and the monitoring unit is installed 10cm above the trash container. If the sensor returns a value of 50cm, the trash level would be:

$$fill\ level = \frac{80 + 10 - 50}{80} \times 100 = 50\%$$

This makes sense, due to the height of the trash level would be 50cm from the sensor, while the sensor is installed 90cm (80+10) from the bottom of the container. Therefore, the current level would be 40cm high, exactly half of 80cm of the trash container, thus 50%.

- Humidity and Temperature Sensor DHT21

The DHT21 is a digital temperature and humidity sensor with a durable, rugged design with a built-in mounting point. It is suitable for this system due to a variety of advantageous features:

- High accuracy and reliability: the DHT21 can measure temperature and humidity with high accuracy, around  $\pm 0.5^{\circ}\text{C}$  and  $\pm 1\%$  relative humidity
- Low power consumption: DHT21 has a low power consumption, making it ideal for applications using batteries as a power source.
- Wide operating range: The DHT21 can measure temperature from  $-40^{\circ}\text{C}$  to  $80^{\circ}\text{C}$  and humidity from 0% to 100% RH.
- Affordable price: the DHT21 sensor maintains a durable design with high data reliability even with an inexpensive price point.



*Figure 6 Humidity and Temperature Sensor DHT21*

The DHT21 works by sending a start signal to the microcontroller. The microcontroller then sends a request signal to DHT21. The DHT21 then sends a response signal that includes relative humidity and temperature information [13].

Relative humidity is measured by a capacitive humidity sensor. The sensor consists of two electrodes with a moisture-holding substrate as a dielectric between them. When the humidity level changes, the capacitance of the sensor changes. The DHT11 measures the change in capacitance and converts it to a relative humidity value.

The temperature is measured by a thermistor. A thermistor is a resistor whose resistance changes with temperature. The DHT21 measures the resistance of the thermistor and converts it to a temperature value.

**Voltage regulator LM2596:**



All the above electronics require a 5V power source to operate. In this design, the power input is chosen to be 3.7V Lithium Ion 18650 batteries connected in series to create a 8V power source. Therefore, a voltage regulator is required to convert 8 down to 5V for the modules to function normally. However, the LoRa SX1278 chip must not be powered by a direct 5V input, as it only works at 3.3V. Thus, the LoRa chip must only be connected to the 3.3V output of the NodeMCU board. The voltage regulator LM2596 is chosen for its small size and ease of operation.



  
*Figure 7 Voltage Regulator LM2596*  
Vietnamese-German University

**Power Source:**

The power source of choice for supplying electricity to the modules above are 18650 batteries connected in series to form an 8V power source, which is then regulated to 5V. Each battery has a capacity between 1200mAh and 3500mAh depending on the model.

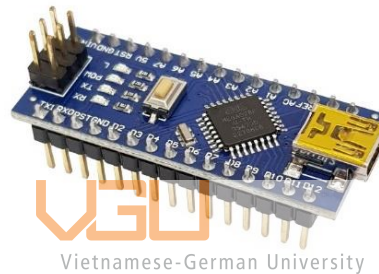


*Figure 8 18650 Li-ion batteries*

**Trash Container:**

- Arduino Nano:

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328P microcontroller (MCU) and developed by Arduino.cc [14]. In this design, the Nano is used as a microcontroller for the trash container, controlling its feature of displaying a notification on an LCD display on the bin, which reminds the user to categorize trash. This feature is implemented by using a human detection sensor, it can detect any human in proximity for a specific duration, and when requirements are met, a warning is displayed. Its small size and ease of programming is the reason it is preferred over other larger microcontrollers. The lack of many input pins is not a disadvantage in this scenario, since the only modules attached to this microcontroller are a sensor and a display unit.



*Figure 9 Arduino Nano*

- Nextion 7-inch display NX8048T070:

To display the notification for users to categorize trash correctly, a display module is installed and connected to the Arduino Nano. The choice of display model here is a balance of size and price, resulting in the model Nextion NX8048T070 due to its large 7-inch screen and ease of connectivity to the Arduino Nano/Uno microcontrollers while having an acceptable price [15]. The Nextion is 800 x 480 pixel display that uses the Human Machine Interface (HMI) interface,. With the high pixel density and large size, information conveyed through the display can be optimized to deliver the message effectively without too many constraints.



Figure 10 Nextion 7-inch display module

- Human Presence Sensor mmWave Rd-03

The Rd-03 is a radar sensor that uses millimeter waves (mmWaves) to detect the presence of humans. Millimeter waves are a type of electromagnetic radiation with wavelengths that are much shorter than those of visible light [16]. They are able to penetrate through objects, such as walls and clothing, which makes them ideal for detecting human presence in a variety of environments. This is suitable for the function where any users in proximity of the trash container would be detected and alerted with a notification shown on the display of the trash container. This particular module is chosen for its balance between reliability and price. Less expensive modules could be inefficient in detecting nearby human, while modules with a higher price point could make the whole system too costly to manufacture.

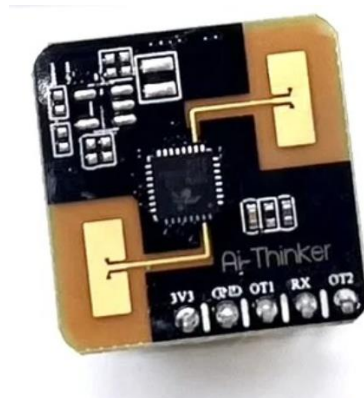


Figure 11 Rd-03 mmWave Human Presence Sensor

### 3.1.3 Bill of Materials:

Since the system proposed has an arbitrary number of nodes, the bill of materials (BoM) for the system will be divided into 4 sub bills: 2 BoMs for the design of the monitoring units (NodeMCU and Heltec boards), 1 BoM for the server unit, and the last one for the components installed in the trash container. These prices exclude the manufacturing of any container for the units, due to a large variety of materials. Furthermore, the design of the enclosures for these unit is subject to change if needed, which affects the manufacturing cost.

BoM for the monitoring unit using NodeMCU:

Part number	Part name	Part description	Quantity	Price per unit	Total price
1	NodeMCU ESP8266 LUA CH340	Microcontroller with Wi-Fi	1	68,000 VND	68,000 VND
2	LoRa Ra02 SX1278	LoRa module	1	140,000 VND	140,000 VND
3	IOE-SR05	Ultrasonic sensor	1	58,000 VND	58,000 VND
4	DHT21	Temperature and Humidity sensor	1	112,000 VND	112,000 VND
5	LM2596	Voltage regulator	1	17,000 VND	34,000 VND
6	Wires	Jumper wires	1	20,000 VND	20,000 VND
7	Batteries	18650 3500mAh batteries	2	145,000 VND	290,000 VND
8	Miscellaneous	Battery casing, screws, etc.	1	40,000 VND	40,000 VND

Table 1 Bill of Materials for NodeMCU monitoring unit

BoM for the monitoring unit using Heltec LoRa ESP32

Part number	Part name	Part description	Quantity	Price per unit	Total price
1	Heltec LoRa ESP32	Heltec development board with LoRa	1	420,000VND	420,000VND
3	IOE-SR05	Ultrasonic sensor	1	58,000 VND	58,000 VND
4	DHT21	Temperature and Humidity sensor	1	112,000 VND	112,000 VND
5	LM2596	Voltage regulator	1	17,000 VND	34,000 VND
6	Wires	Jumper wires	1	20,000 VND	20,000 VND
7	Batteries	18650 3500mAh batteries	2	145,000 VND	290,000 VND
8	Miscellaneous	Battery casing, screws, etc.	1	40,000 VND	40,000 VND

Table 2 Bill of Materials for Heltec LoRa ESP32 monitoring unit



Vietnamese-German University

BoM for the server unit:

Part number	Part name	Part description	Quantity	Price per unit	Total price
1	Heltec LoRa ESP32	Heltec development board with LoRa	1	420,000VND	420,000VND
2	LM2596	Voltage regulator	1	17,000 VND	34,000 VND
3	Wires	Jumper wires	1	20,000 VND	20,000 VND
4	Batteries	18650 3500mAh batteries	2	145,000 VND	290,000 VND
5	Miscellaneous	Battery casing, screws, etc.	1	40,000 VND	40,000 VND

Table 3 Bill of Materials for server unit

BoM for the components of the trash bin:


Part number	Part name	Part description	Quantity	Price per unit	Total price
1	Arduino Nano	Heltec development board with LoRa	1	89,000 VND	89,000 VND
2	LM2596	Voltage regulator	1	17,000 VND	34,000 VND
3	Nextion 7-inch display	Display	1	1,500,000 VND	1,500,000 VND
4	Batteries	18650 3500mAh batteries	2	145,000 VND	290,000 VND
5	Miscellaneous	Battery casing, screws, etc.	1	40,000 VND	40,000 VND
6	Wires	 Vietnamese-German University	1	20,000 VND	20,000 VND

Table 4 Bill of Materials for components in the trash container

### 3.1.4 Processes

Before the assembly of all above components, a process for each procedure of the system to ensure smooth and precise operation. The system is divided into 4 processes, the first of which is data monitoring, collection, and transfer to the server of the trash containers. The next process is signal processing and data upload to the Internet by the LoRa server. Third, the data uploaded to the server must be monitored and exported to an application to generate a suitable short waste collection route. Finally, while not playing a major role in data collection and route generation, there must also be a process for the trash containers where it displays notifications to any users standing in front of the trash bins for an extended period of time.

The flow chart for the first process is illustrated in Figure 12. After program start, the monitoring unit establishes LoRa communication at a specific frequency, then reads the data from the unit's sensors. After the data is validated, it is sent via LoRa to the frequency selected for the server to listen to and pick up. The unit will continually send data for 5 minutes to ensure no loss data transmission to the server, after which it will go to sleep for 55 minutes to save battery power. The process is then repeated after the unit is woken up, which means new data is updated every hour. Additionally, the unit can be configured to go into standby at night to further save energy.

The flow chart for the LoRa server that receives data from multiple monitoring units is described in Figure 13. When the program starts, the server begins LoRa communication and actively listen for signals sent at a specific syncword/address and frequency, then decipher the data sent, which is written in a specific format. If the data deciphered is valid, it is mapped to the respective variables corresponding to the original monitoring unit's device number. Afterward, the values are then uploaded to the Internet to an IoT platform, ThingsSpeak.

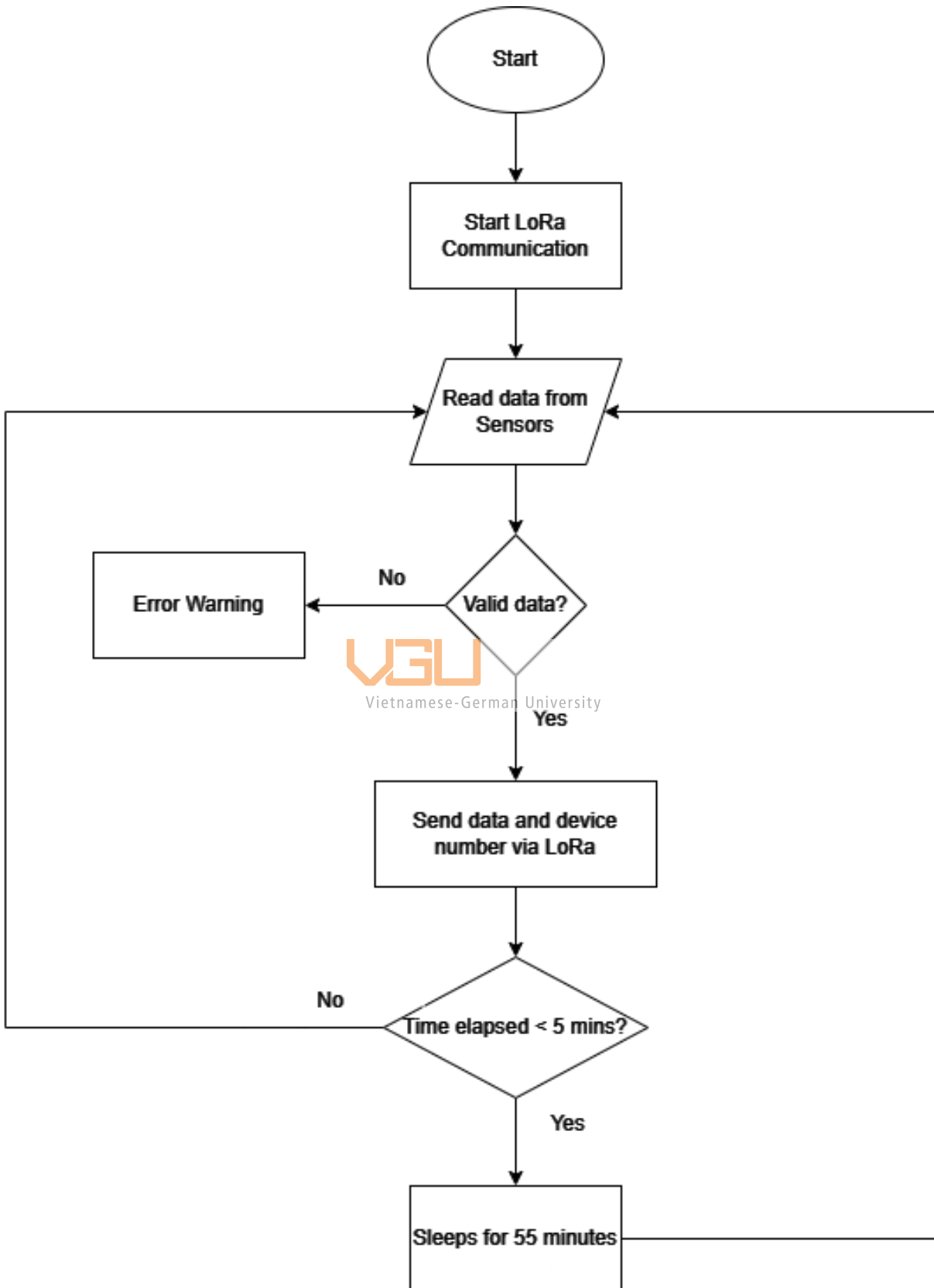


Figure 12 Data Collection Flow Chart



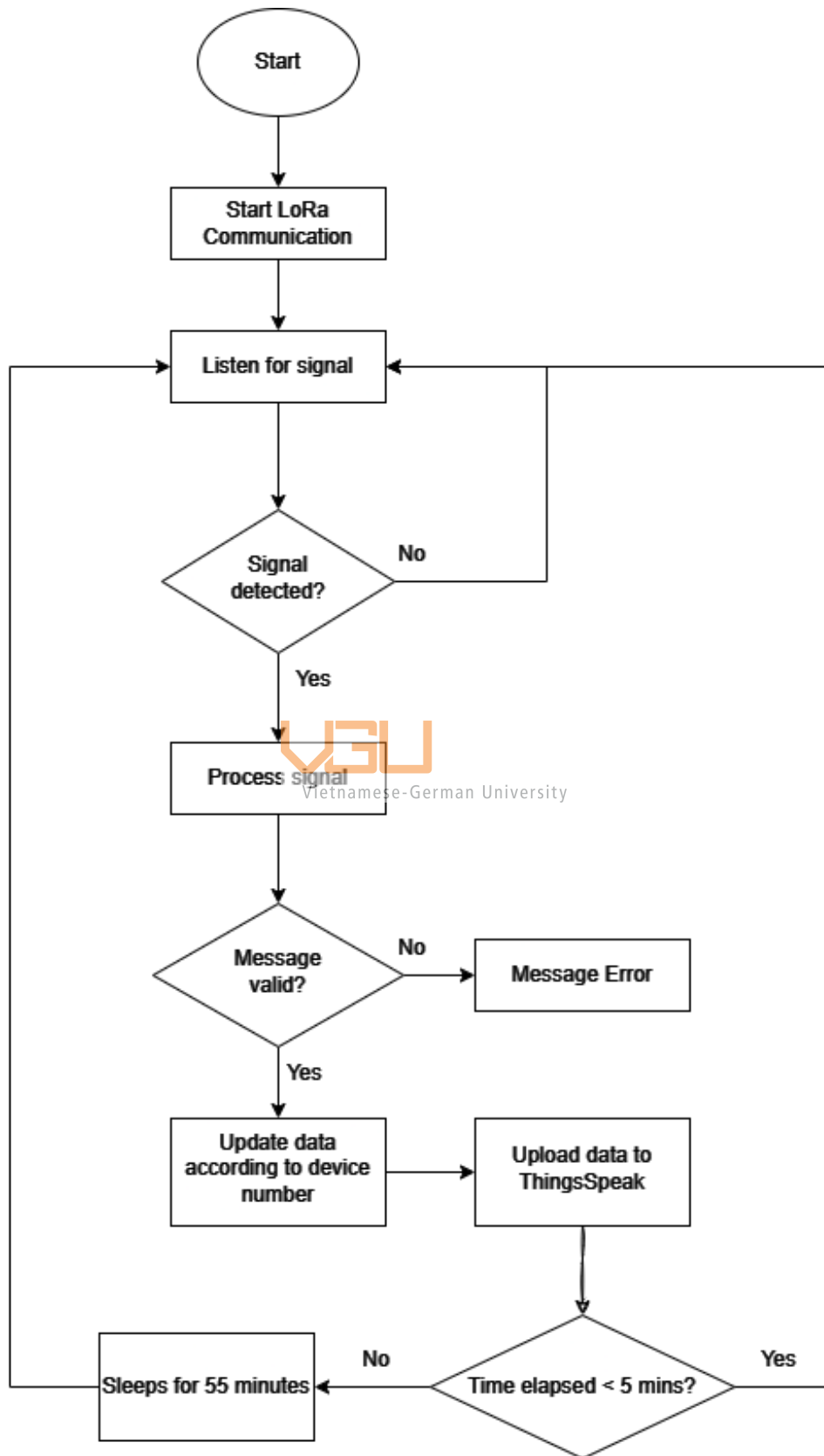


Figure 13 LoRa Server Flow Chart

The process to determine the appropriate algorithm to generate the shortest waste collection route is depicted in the flow chart in Figure 14. Since the number of full bins changes depending on their states, the system must read the exported data to count the number of nodes, after validating that there is no abnormalities in the data. With fewer nodes, a guaranteed shortest path can be achieved by using either brute force or branch and bound algorithm without much time spent, while with a sufficiently large number of nodes where most algorithms would take too long to process, the nearest-neighbor method is better at giving a reasonable result. Otherwise, an acceptable but not shortest route can be generated by many other algorithms, after comparing multiple iterations of those algorithms to determine the shortest one.

Finally, because the trash containers feature the ability to display a notification to remind the users to categorize waste, the process is illustrated in Figure 15. The installed proximity sensor can detect a user near the trash container, and if they stay for an extended period of time, which is chosen to be 5 seconds here, the display on the container will display a message to remind the user which type of trash belong to their respective categories, being organic and inorganic. The time limit 5 seconds is chosen so as to avoid random activations when other users pass by the container.

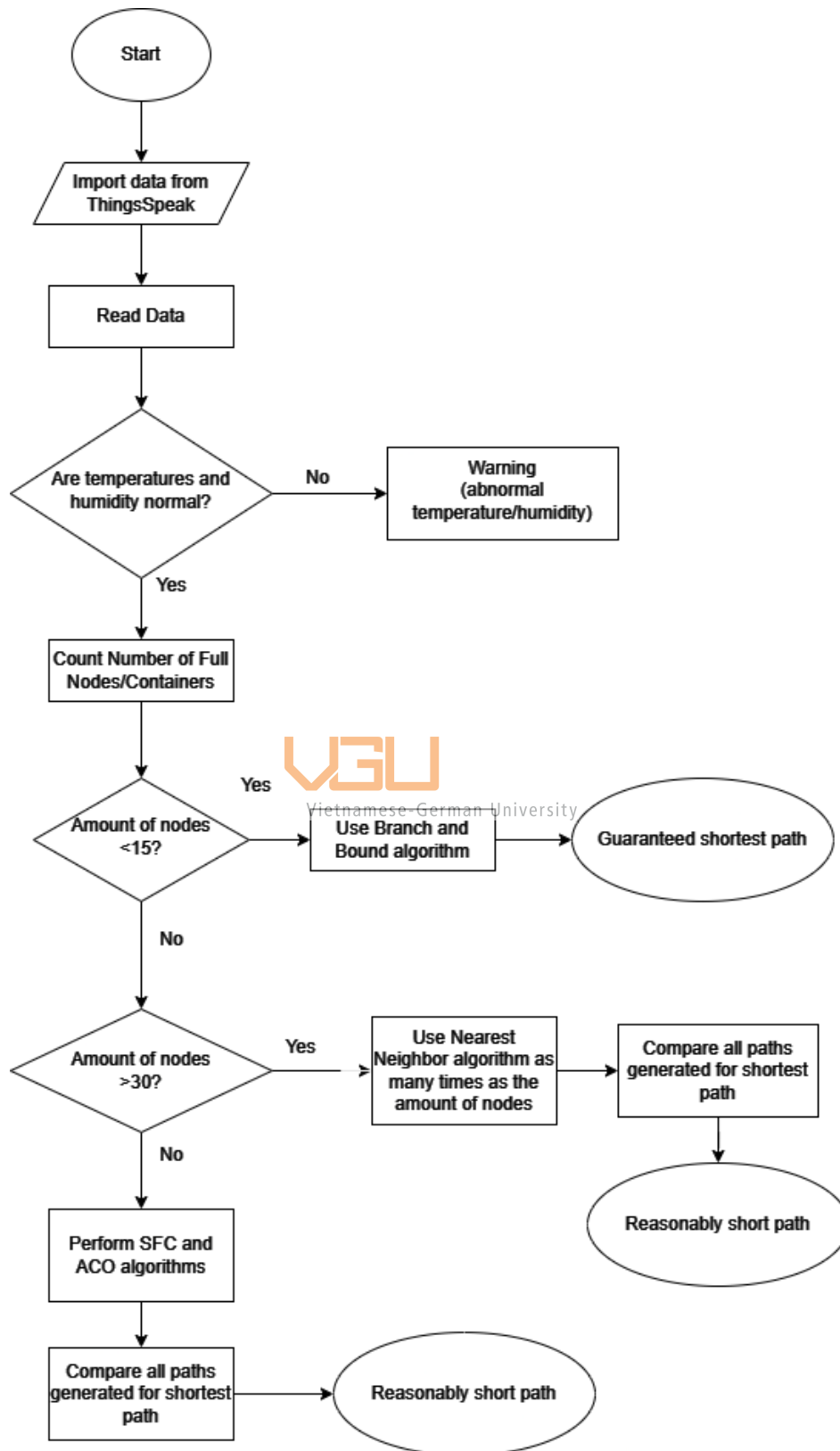


Figure 14 Shortest Route generation Flow Chart

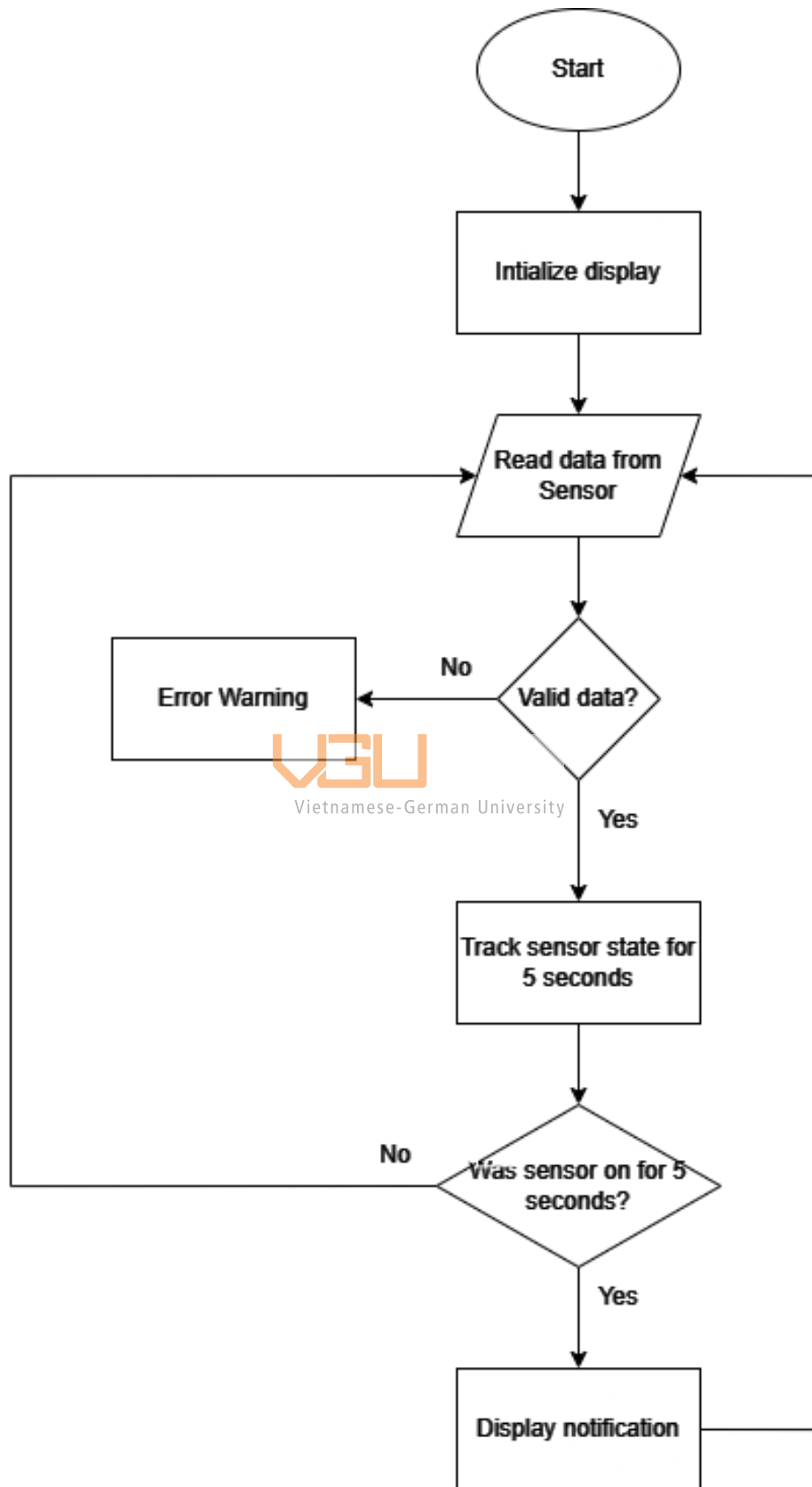


Figure 15 Trash container User Notification Flow Chart

## 3.2 Technical Design

### 3.2.1 Trash container design

The design of the trash container is split into three parts: the upper compartment where the monitoring unit is attached, and where the detection sensors and the display is installed. The second part is a simple trash bin for storing waste. Finally, a dock holds two bins together to create the final assembly. In front of the bins are recessed panels that can be used to engrave additional information on the categories of trash.

The design choice to incorporate a dock and separating the container into two bin serves several purposes: more convenient maintenance of the bins, ease of trash collection, a more distinctive design to further point out the two categories of trash, and lastly, this design makes the container more modular, any unit in the system is interchangeable in terms of components.

The recommended material for this container design is stainless steel for the dock and the bins to enhance their durability, while having a uniform look. The upper part where electronics are stored can be made of both stainless steel and other composite materials, for better distinguishment from the two bottom parts, drawing attention of the users.

The whole container measures 1010mm, or over 1m in width, 500mm or 0.5m in length, and 1m in height. The bin is 800mm tall and has a rounded square cross section with 440mm sides, thus having the volume of  $0.8 \times 0.4 \times 0.4 = 0.128 \text{ m}^3$ . This is suitable for placement around a university campus, since the average amount of trash disposed daily per Vietnamese person is around 1.2kg [17]. Therefore, the bins can realistically hold up to 3 to 4 days of waste if many bins are installed around the campus, eliminating the need to empty out the bins every single day.

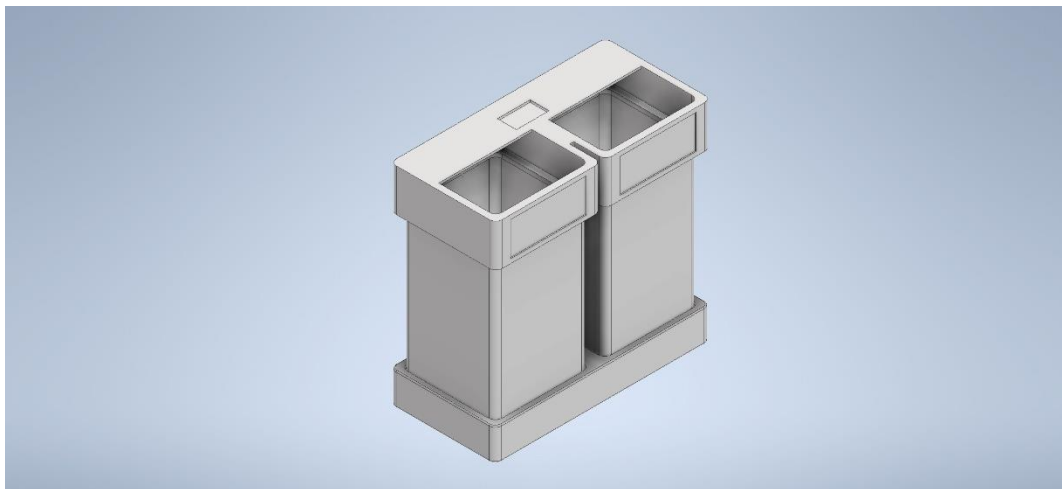
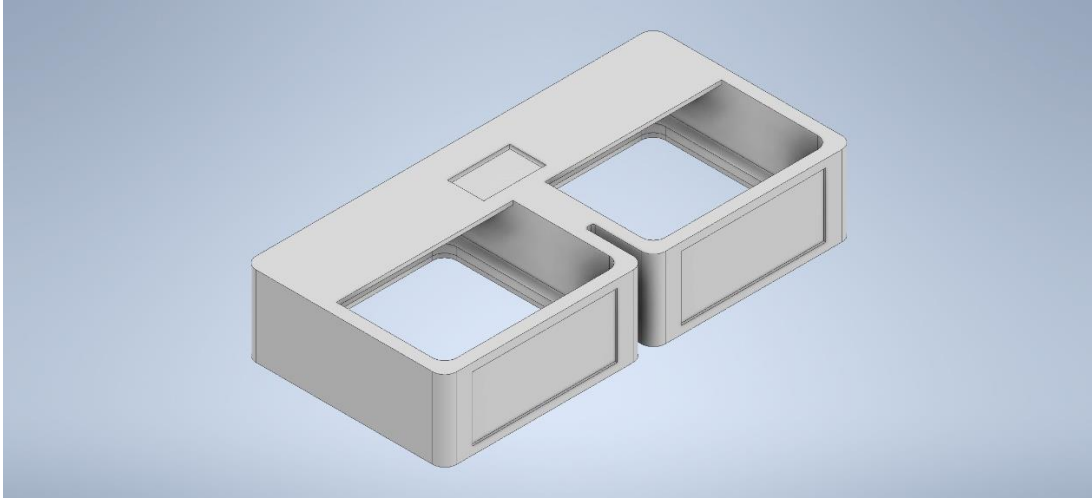


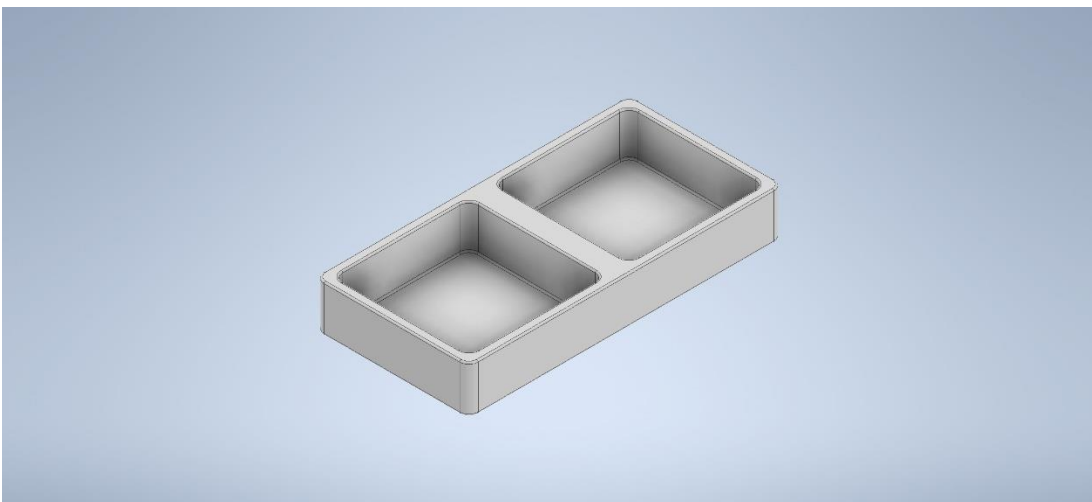
Figure 16 The assembly of the trash container



*Figure 17 The upper part of the trash container*



*Figure 18 The bin of the trash container*



*Figure 19 The dock of the trash container*

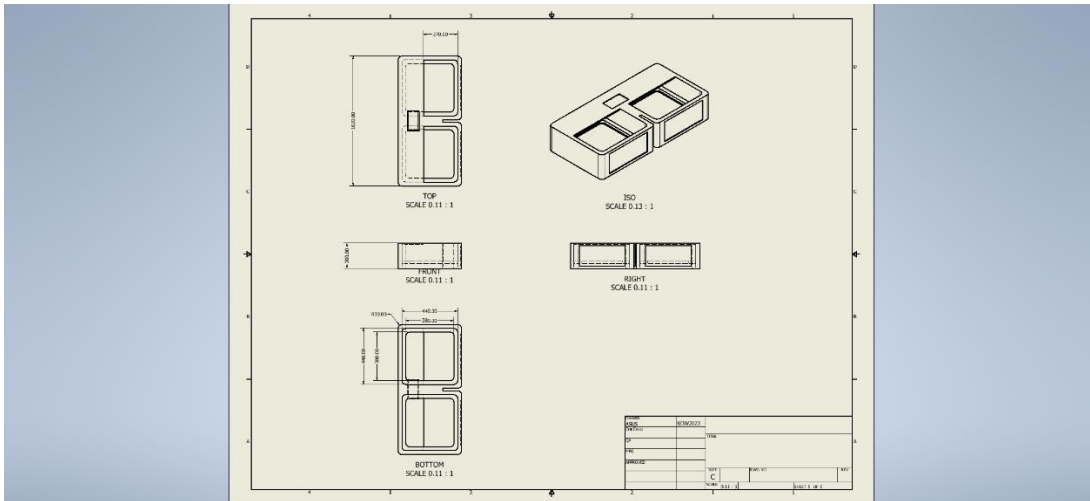


Figure 20 Drawing of the upper part of the trash container

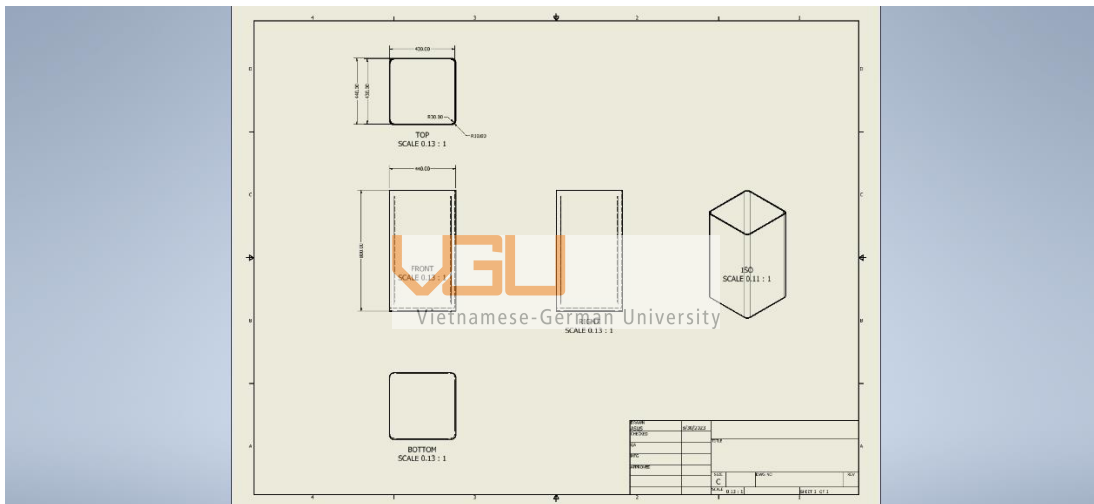


Figure 21 Drawing of the bin of the trash container

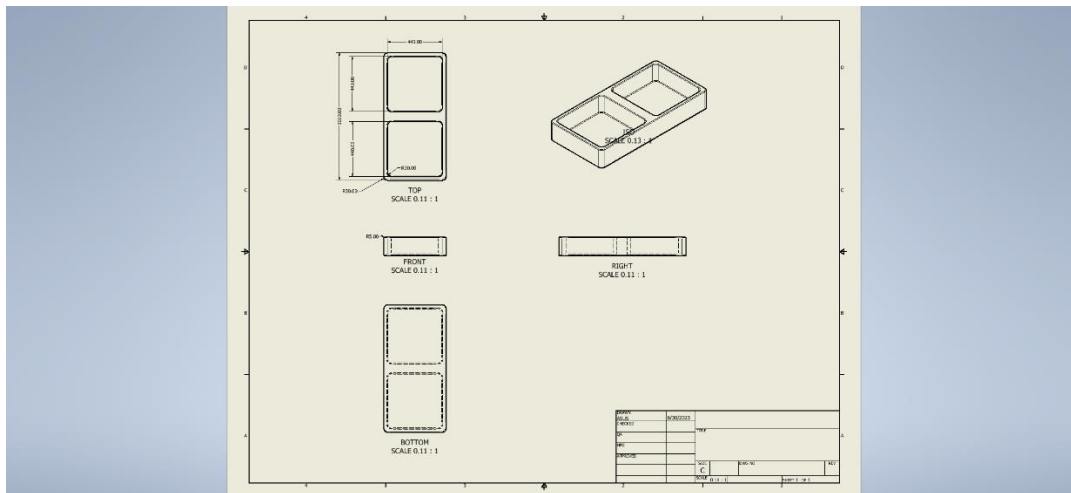


Figure 22 Drawing of the dock of the trash container

### 3.2.2 Design of the monitoring unit enclosure

Designing the enclosure for the monitoring unit is more nuanced: there needs to be careful consideration for all components' dimensions and a balance between size and practicality. Too much space for installation comfort can lead to an overly large design that can affect the mounting of the enclosure to the trash container. An enclosure too small and the wiring and installation of all electronics can be difficult to perform.

Close inspection reveals that the NodeMCU ESP8266 board combined with the LoRa module SX1278 yields similar dimensions to the all-in-one Heltec Wi-Fi LoRa board, therefore the same enclosure design can accommodate both microcontrollers, since all the other components are shared between the two: sensors, voltage regulator, power source, antenna and even the placement of their USB ports. The only difference between the two enclosures would be the mounting point for the microcontrollers, which have different radii and positions.

The top cover of the enclosure is fitted with a rubber gasket to help protect the inside components from moisture damage. The mounting point to the trash container is offset after taking into account the size and weight of the power source, which is mounted externally for ease of replacement and powering on/off the device. After assembly, the enclosure measures 101mm in length, 75mm in width and 43mm in height. Below are images of the design for all the monitoring enclosures of the proposed waste management system:

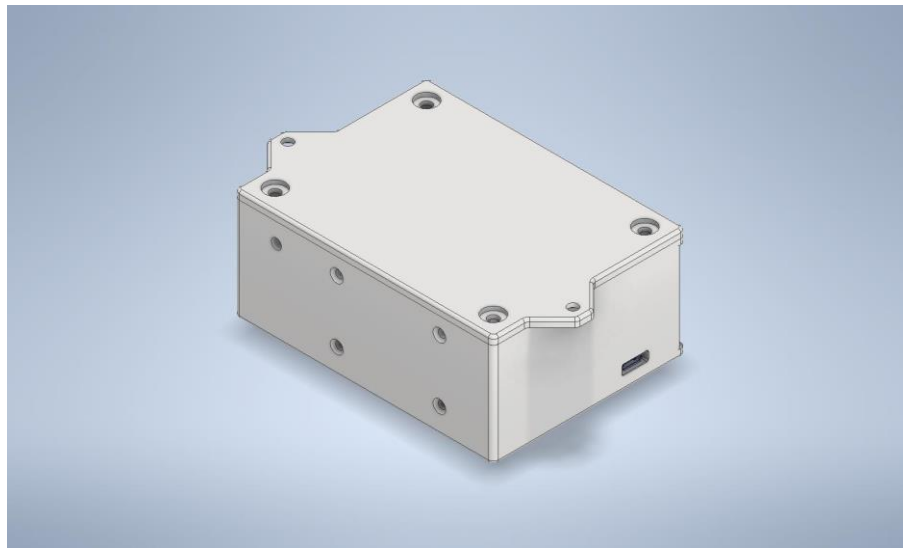


Figure 23 Assembly of a monitoring unit



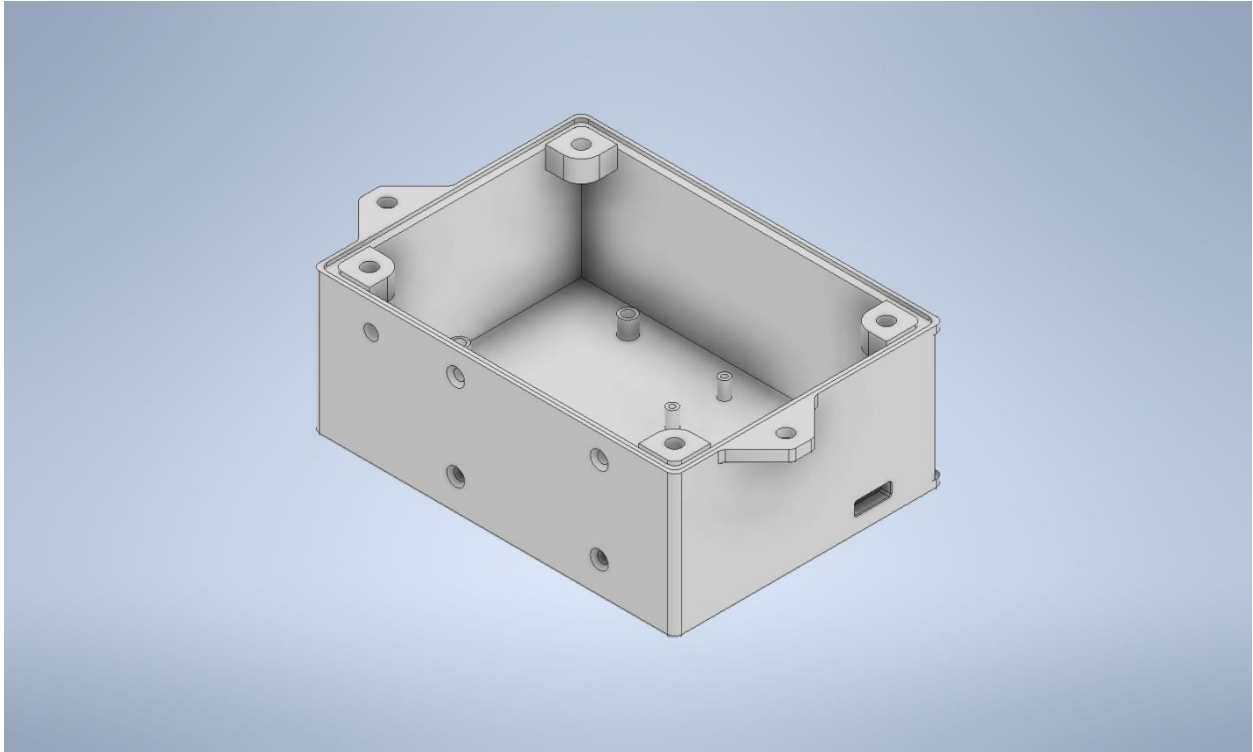


Figure 24 The Heltec Wi-Fi LoRa enclosure, with smalling mounting points

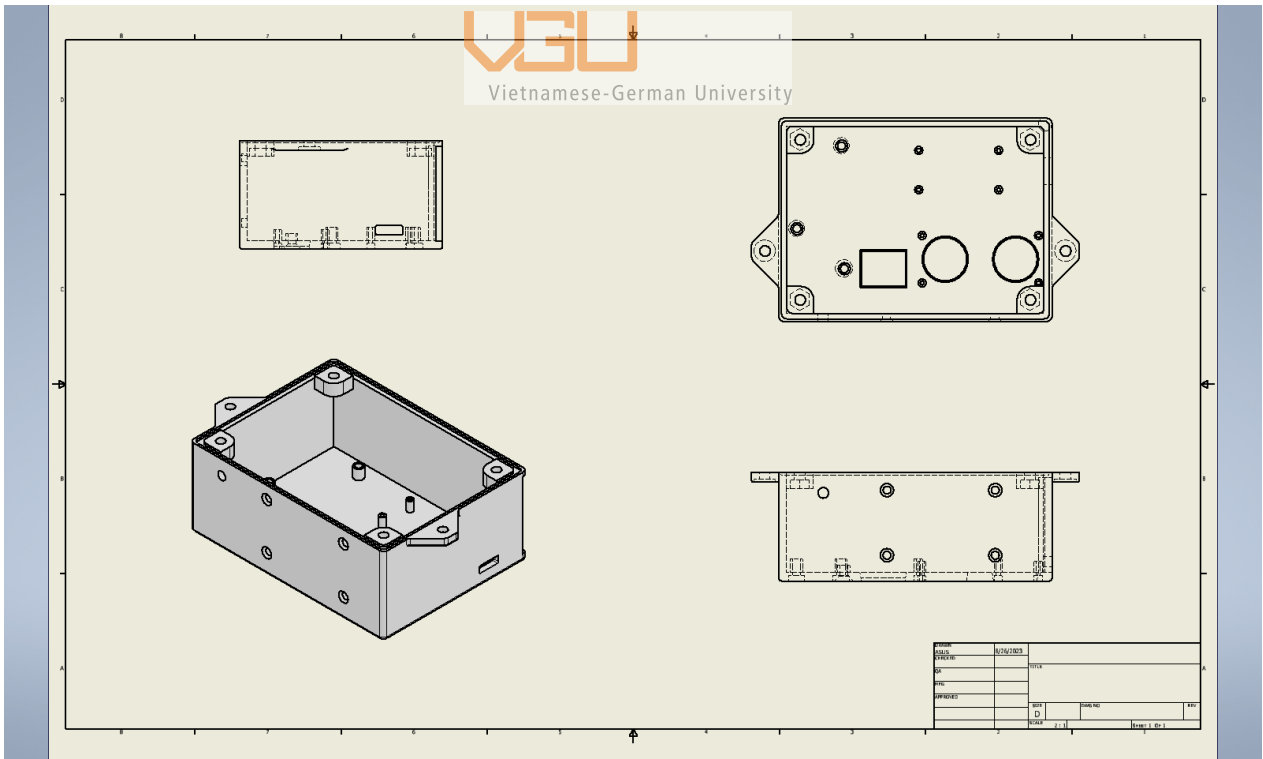


Figure 25 Drawing for the Heltec Wi-Fi LoRa enclosure

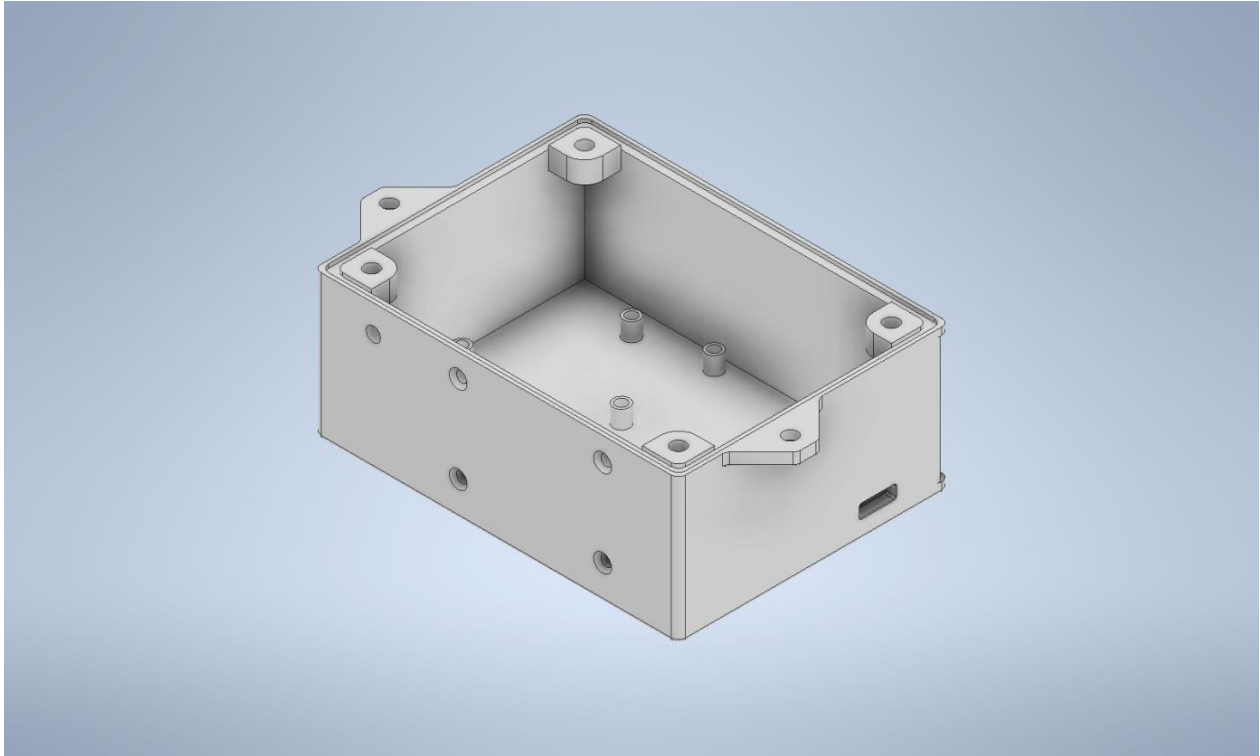


Figure 26 The NodeMCU enclosure, with larger mounting points

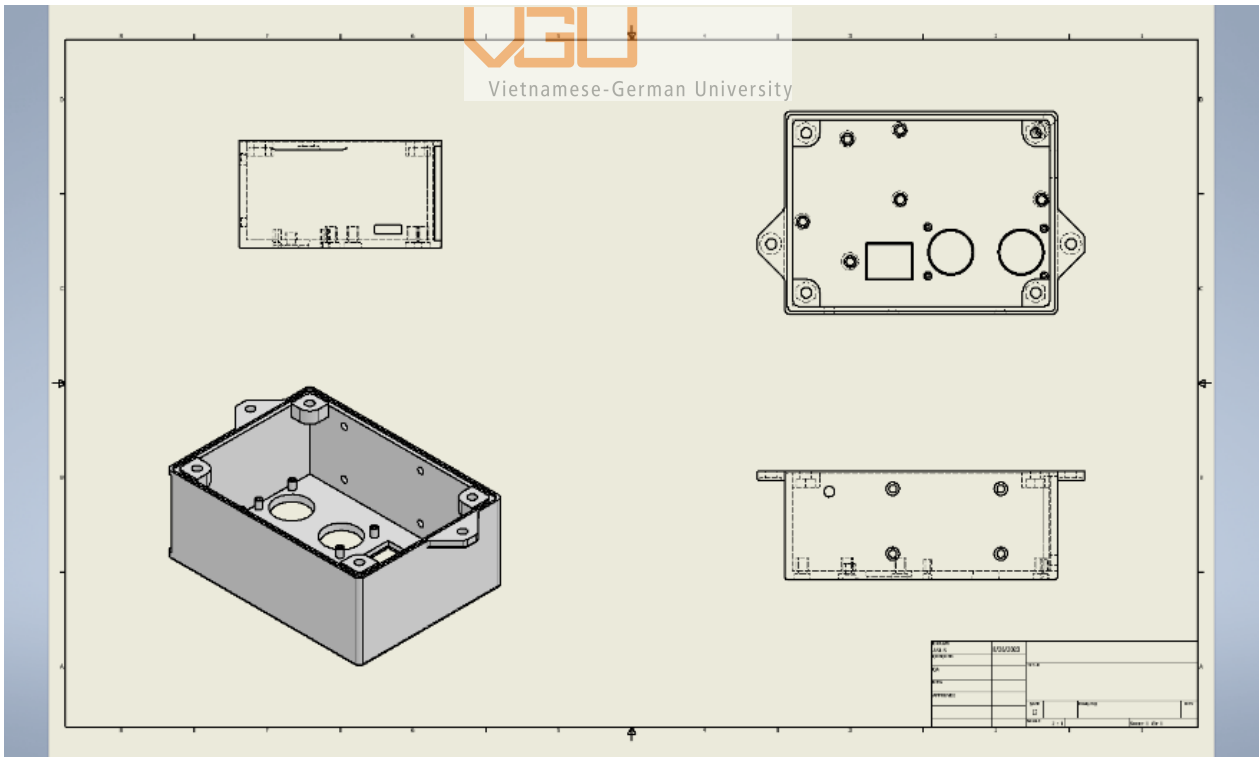


Figure 27 Drawing for the NodeMCU enclosure

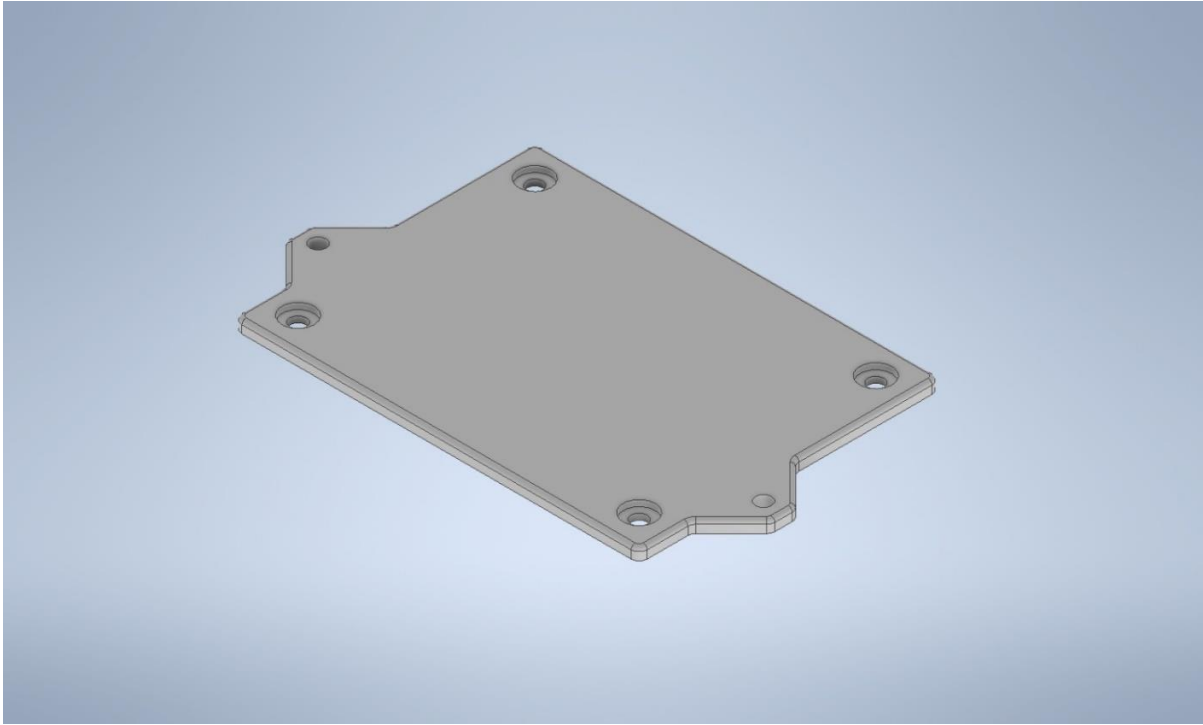


Figure 28 View of the top cover for the enclosures

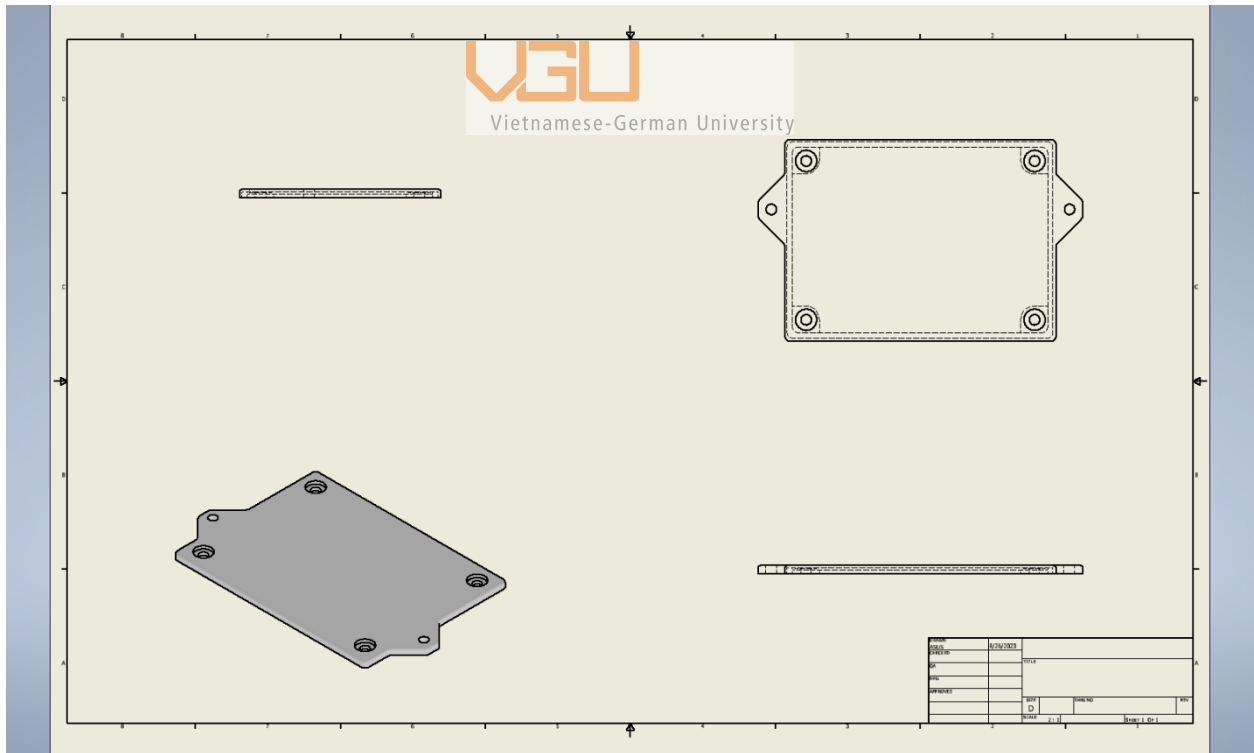


Figure 29 Drawing of Top cover

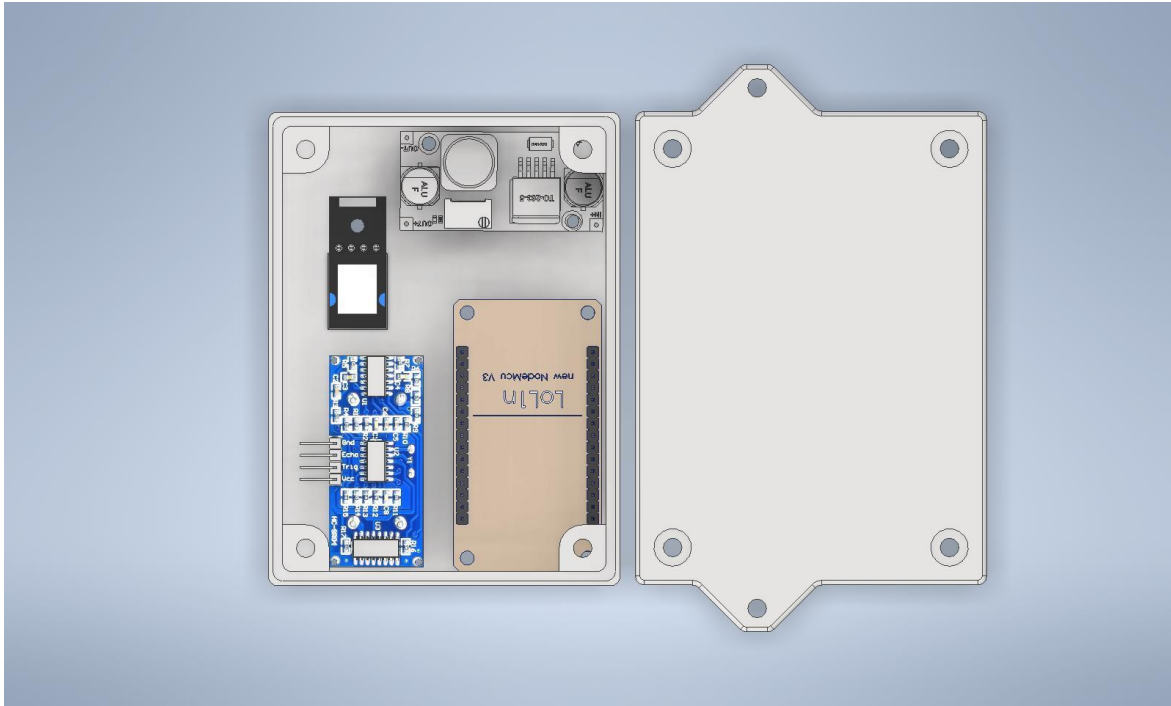


Figure 30 A preview of components installed in a monitoring unit

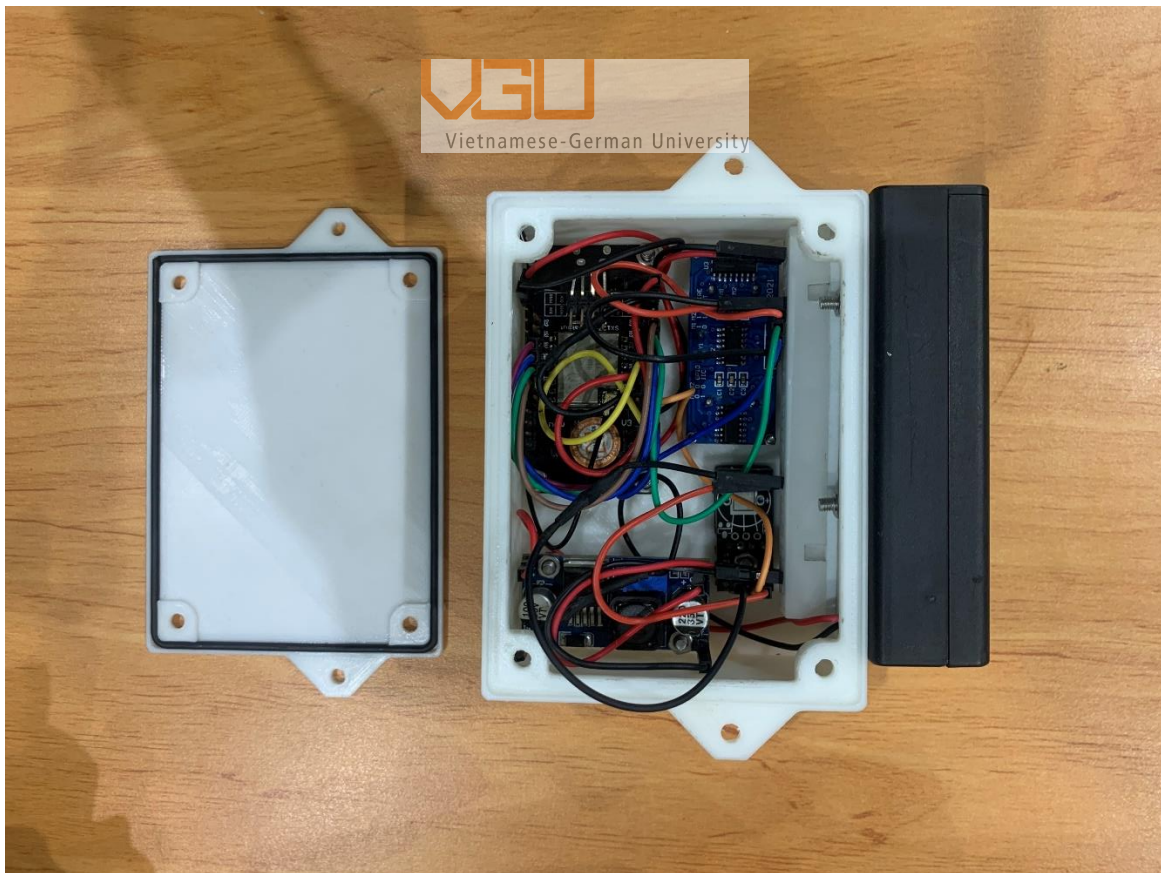


Figure 31 The experimental assembly of the monitoring unit

### 3.3 Algorithms

#### 3.3.1 Definition of the problem

There are many algorithms that involve finding the shortest paths through a graph, such as Dijkstra's algorithm, Floyd-Warshall, etc. However, the problem presented in finding an optimal route to collect trash from full bins around a campus is different from just finding the shortest path; hence, the mentioned algorithms are not quite applicable. Said problem rather resembles the well-known Traveling Salesman Problem (TSP), which states that a salesman must travel through all given city only once and then return to his starting point via the shortest path.

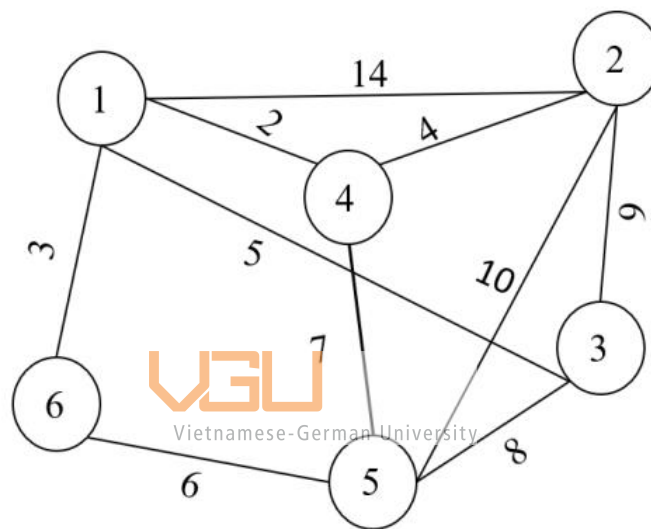


Figure 32 An example Traveling Salesman Diagram

#### 3.3.2 Solutions to the problem

There are many solutions to the Traveling Salesman Problem that can be applicable to find the best path, or near-best path available. Here are a few that shall be discussed:

- Brute force method.
- Nearest-Neighbor method.
- Branch and bound method.
- Space-Filling Curve method.
- Ant colony optimization method.

#### 3.3.3 The brute force method

As the name suggests, the brute-force method tries all the possible paths and compares them to find the shortest path available. It works by generating all possible solutions to the problem and

then testing each solution to see if it is a valid solution. If the solution is valid, the algorithm returns the solution. If the solution is not valid, then the algorithm discards the solution and moves on to the next solution [18]. The brute-force method is then expressed by the algorithm:

```
def find_shortest_route(routes):
    shortest_route = route[0]
    for route in routes:
        if route < shortest_route:
            shortest_route = route
    return shortest_route
```

This algorithm works by first setting the shortest route to the first route in the list. Then, it iterates through the list of routes, comparing each number to the shortest route. If the route is shorter than the current shortest route, then the shortest route is updated to the new route. The algorithm returns the shortest route at the end.

However, the more nodes and edges added to the graph, the more complex the problem will be and the longer the algorithm will take to go through all iterations. To be exact, assume that all nodes are connected; for a number  $n$  given nodes, there are  $(n - 1)$  choices to start from node 1 to the remaining nodes. Afterward, from the second node, there are  $(n - 2)$  choices to proceed to the third node. Therefore, to reach the end node there are  $(n - 1)!$  possible paths, and since the end node and the first node connect, the result is  $\frac{(n-1)!}{2}$  iterations. That means that for just 20 nodes, there are quadrillions of possible paths, which will take a very long time for a computer to process. Therefore, while this approach is guaranteed to give the shortest route, it will take too long to process if given a realistic scenario (a smart trash bin system for a large university campus, including its dormitories, will have more than 20 bins). Therefore, it should only be implemented in the case of less than 15 nodes, for example, in summer where there are fewer students on campus, the number of full bins should decrease, hence creating fewer nodes in the collection route). Thus, the system generating the shortest path should first consider the number of full nodes to decide whether to execute the brute force method first for an exact solution to the shortest route or to try other methods more suitable for a larger set of nodes.

### 3.3.4 Branch and bound method

The branch-and-bound method is a recursive algorithm to find the optimal solution to an optimization problem. It works by constructing a tree of possible solutions and then exploring the tree in a way that ensures that the optimal solution is found eventually [19]. To solve the TSP using branch and bound, we start by creating a tree with one node for each possible way to start the tour. Each node in the tree has  $n-1$  children, one for each city that the salesman could visit next. We then explore the tree, one node at a time. For each node, we first compute a lower bound on the cost of the optimal tour that starts at that node. This lower bound is an estimate of the minimum possible cost of a tour that starts at that node and it is used to prune the tree. If the lower bound for a node is greater than the cost of the best tour found so far, then we know that the node cannot contain the optimal solution and we can discard it without further exploration. We continue exploring the tree until we reach a leaf node, which represents a complete tour. The cost of the tour represented by the leaf node is then compared to the cost of the best tour found so far. If the cost of the leaf node is less than the cost of the best tour found so far, then the leaf node represents the new best tour.

The branch-and-bound method is guaranteed to find the optimal solution to the traveling salesman problem, but it may not be the most efficient algorithm. The time complexity of the branch-and-bound method depends on the number of possible tours, which can be exponential in the number of cities. Below is an example of how the branch-and-bound method can be used to solve the traveling salesman problem for a set of 4 cities.

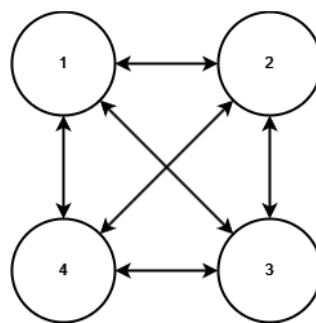
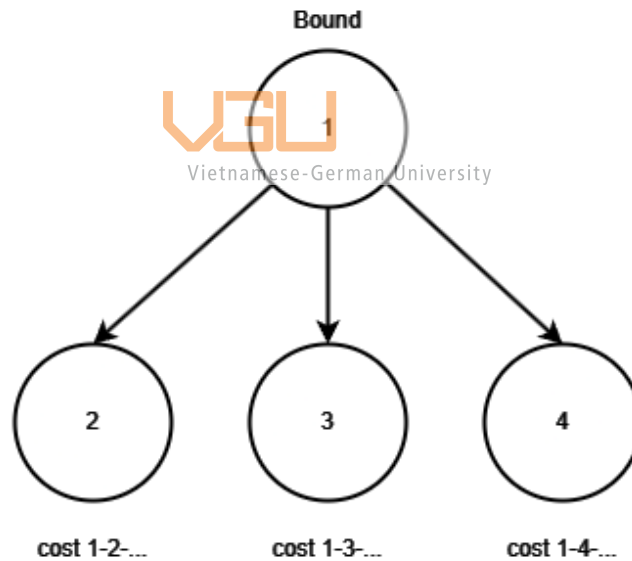


Figure 33 Example 4 nodes

1. Initially, the tree has one node, which represents the start of the tour. (node 1)
2. The lower bound for this node is the cost of the shortest route from the starting city to any other city.

3. The children of this node are one for each of the other cities.
4. The lower bound for the child node that represents the city with the shortest path is the cost of the shortest path from the start city to that city plus the lower bound for the child node.
5. The lower bound for the child node that does not represent the city with the shortest path is the cost of the shortest path from the start city to that city plus the cost of the shortest path from that city to the city with the shortest path.
6. We continue exploring the tree in this way until we reach a leaf node, which represents a complete tour.
7. The cost of the tour represented by the leaf node is then compared to the cost of the best tour found so far.
8. If the cost of the leaf node is less than the cost of the best tour found so far, then the leaf node represents the new best tour.



*Figure 34 Branch-and-bound example*

The branch-and-bound method can be used to solve the TSP for any number of cities. However, the time complexity of the algorithm increases exponentially with the number of cities. Therefore, the branch-and-bound method is only practical for solving the traveling salesman problem for small- to medium-sized problems.



### 3.3.5 Nearest-Neighbor method

The nearest-neighbor method examines the edges that connect to the starting node and chooses the shortest edge to traverse to the next node. The process then repeats, finding the nearest node from that node and going to it, until the last node is reached, then it simply returns to the starting node to complete the problem [20]. While this seems intuitive, it is not guaranteed that the optimal route is found, even if we choose different starting points to compare, since sometimes the optimal path is found by sacrificing a nearest node to come to another node that is otherwise too far to travel to from any other nodes.

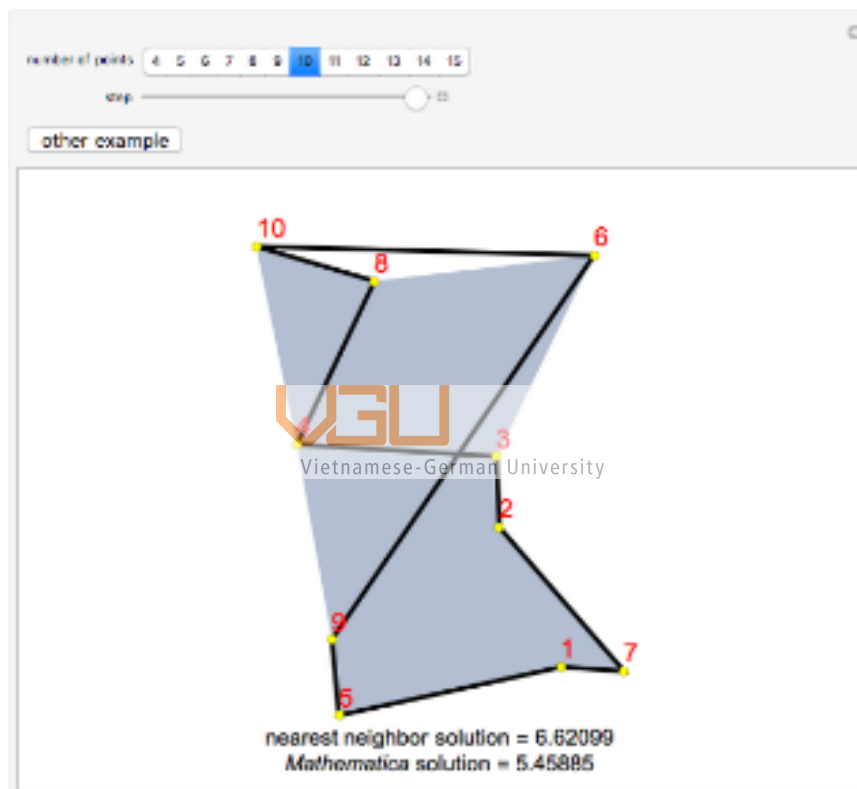


Figure 35 Nearest-neighbor method example

### 3.3.6 Space-Filling Curve Method

The space-filling curve method imposes a grid of squares onto the graph such that each node is isolated in a square with no other nodes. Then, a space-filling curve is added that goes through all the squares, marking the order of the nodes as it fills the grid. The nodes are then connected to form a route that goes through all nodes. This is a heuristic approach, which means that it does not guarantee finding the optimal solution, but it can find good solutions in a reasonable amount of time [21].

The following are the steps on how to use a space-filling curve to solve the TSP:

- Generate a space-filling curve for the set of points.
- Sort the points according to their order on the space-filling curve.
- Start at the first point and visit the points in sorted order.

The space-filling curve can be any type of space-filling curve, but some common choices include the Hilbert curve, the Peano curve, and the Sierpinski curve. The choice of space-filling curve will affect the quality of the solution found. Therefore, if there are more nodes than what the brute force and Branch method can handle, multiple curves can be drawn to compare for a good enough solution to the shortest collection route.

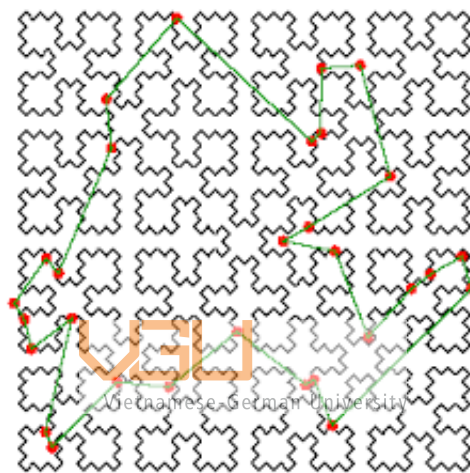


Figure 36 Space-Filling Curve Example

### 3.3.7 Ant Colony Optimization

The Ant Colony Optimization (ACO) method draws inspiration from the behavior of ants in nature. Ants have a natural tendency to find the shortest path between their nest and a food source [22]. Naturally, ants release pheromones when they travel, which helps other ants to either travel back to their nest or to a food source. The pheromones can also evaporate over time and not last indefinitely. With this information, we can observe the way ants find the shortest path when foraging for food, by analyzing the paths the ants take. Ants that travel along a shorter path can leave larger trace of pheromones in the same amount of time compared to ants that have to travel a longer path. Over time, the pheromones left on the shorter path becomes stronger, while pheromones on the longer paths fade and thus attract less ants. With many ants traveling the shorter path, the choice is enhanced and the shortest path is guaranteed.

The ACO algorithm implement a similar approach to reconstruct the method used by the ants. First the algorithm create several ants and assign each a start node and a list of all other nodes. Afterward the ants roam freely around the map to explore nearby nodes, depositing pheromone trails as they travel. After this first iteration, the ants stop and time is allowed to pass, thus evaporating some of the pheromone trails. After this, the ants are allowed to explore the graph again, but this time they are influenced by the pheromone trails left by the first iteration, therefore reinforcing shorter paths by having more ants travel those paths. This process is repeated indefinitely until an optimal path is found [23].

The ACO algorithm for solving the TSP can be explained in detail below:

1. Initialize the colony of ants and create a graph of the nodes.
2. Each ant starts at a random node.
3. The ant chooses a neighbor node to visit based on the following probability:  
probability = pheromone trail / (pheromone trail + distance)
4. The ant deposits a pheromone trail on the path between the current node and the neighboring node.
5. The ant repeats steps 3 and 4 until it has visited all the cities.
6. The ants are then allowed to stand by and the pheromone trails evaporate.
7. The process is repeated until the simulation í, and the shortest path so far is recorded.

The mathematics behind ACO is based on the following principles: Let  $G = (V, E)$  be a graph, where  $V$  is the set of vertices and  $E$  is the set of edges. Let  $\tau(e)$  be the pheromone trail on edge  $e$ .

Let  $d(v, w)$  be the distance between vertices  $v$  and  $w$ .

- Initialize the pheromone trails on all edges to a constant value.
- Create a colony of ants.
- For each ant  $a$ :
  1. The ant starts at a random vertex  $v$ .
  2. The ant chooses a neighbor vertex  $w$  to visit based on the following probability:  
probability =  $\tau(v, w) / (\tau(v, w) + d(v, w))$ .
  3. The ant deposits pheromone  $\tau(v, w)$  at the edge  $(v, w)$ .
  4. The ant repeats steps 2 and 3 until it has visited all vertices.
- Evaporate the pheromone trails on all edges.
- Repeat steps 3 and 4 until a solution is found.

After the ants have performed step 4 and the pheromone trails are partially evaporated, the pheromone value on each edge is now different from each other, unlike at the start, where the pheromone trails on all edges are set to a constant value. Therefore, the probability of the ant choosing a path will be different from the 1<sup>st</sup> iteration, even if it starts at the same node. Through several iterations, the algorithm will eventually find the shortest path possible through all nodes. While for a large number of nodes there is no easy way to confirm if the path found by the ACO algorithm is the guaranteed shortest path, the results from this algorithm are usually better than the above-mentioned methods, while not taking too much time to compute. Therefore, it is the preferred choice for finding the optimal paths to collect waste for a smart waste management system.

Below is an illustration of the ACO algorithm from a Python program by Auctux on Github [24]

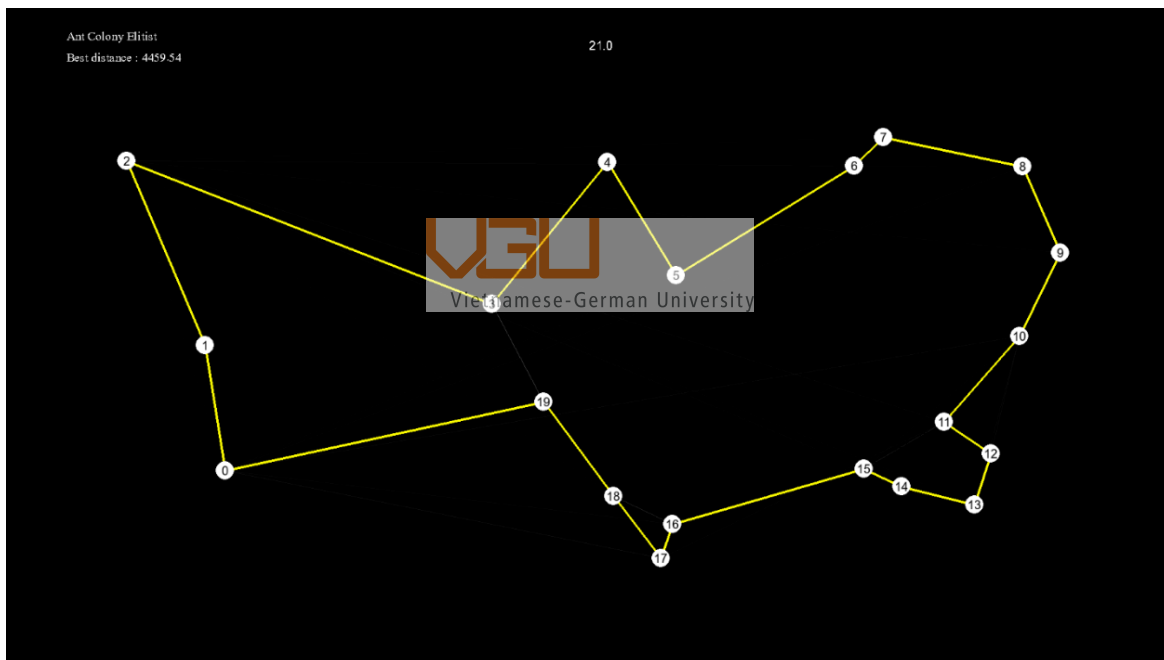


Figure 37 Ant colony optimization example

### 3.3.8 Trial runs

Several trials with the Branch and Bound algorithm can be run using several open-source projects. Here the free calculator by AndrewB330 is chosen for its simplistic design and ease of input [25]. Data input is in by double clicking the interface, the points created can be dragged around to change their coordinates, thus changing the costs to travel between them. A branch and bound visualizer is also included next to the route generation, showing how the system came up with the path.

### 1<sup>st</sup> trial: 5 nodes

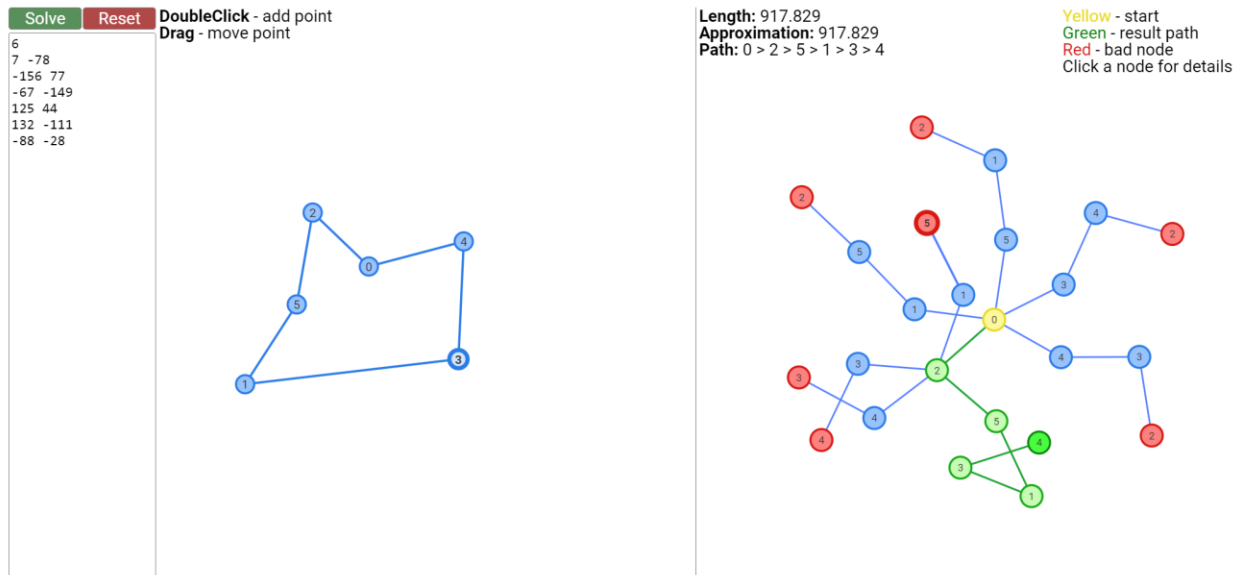


Figure 38 TSP-Solver 6 nodes

For 6 nodes, the algorithm took only 2 seconds to generate the guaranteed shortest path.



### 2<sup>nd</sup> trial: 11 nodes

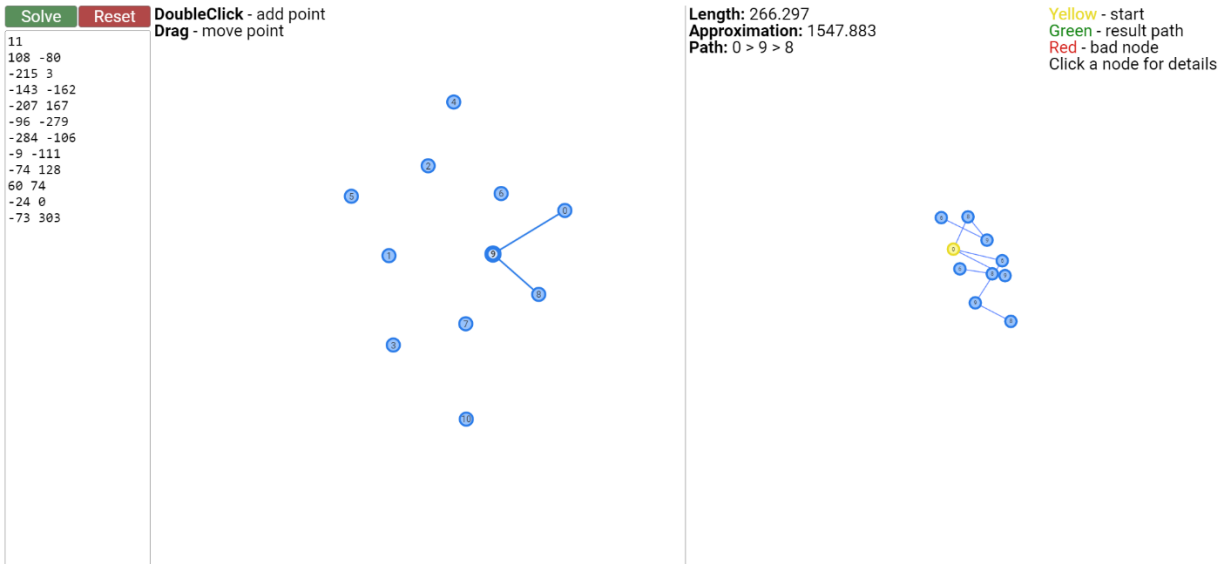


Figure 39 The TSP-Solver running

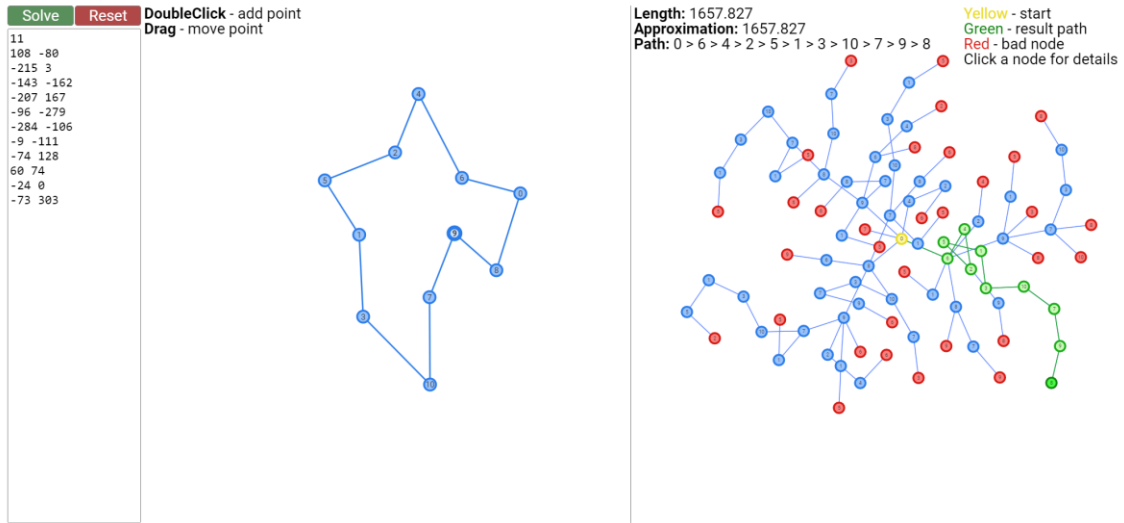


Figure 40 Result from the TSP-Solver

The algorithm took 7 seconds to return the optimal result with 11 nodes. Branches where the path's cost exceed the bound was pruned, as displayed in the simulation. Increasing the number of nodes by two times resulted in nearly quadruple the computing time, proving the point made earlier that the algorithm, while guaranteed to return the shortest path possible, would increase in complexity and unable to return a good result within a reasonable amount of time if the number of nodes were to be to many.



Vietnamese-German University

### 3<sup>rd</sup> trial: 12 nodes

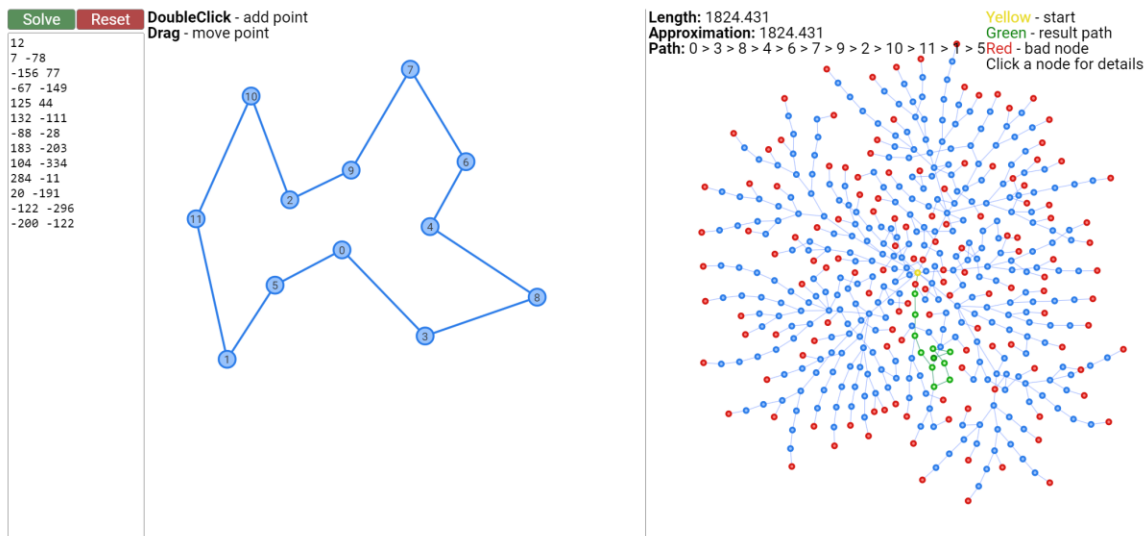


Figure 41 TSP-Solver 12 nodes

With the addition of the 12<sup>th</sup> node, the computation time of the algorithm went up to 33 seconds, over 4 times the time to compute 11 nodes. This further proves that the more nodes there are, the computation time for this algorithm will increase drastically, hence unable to deliver a guaranteed result in a short amount of time.

### 3.3.9 Conclusion

Each of the described methods has its advantages and disadvantages, therefore the methods used should be based on how many nodes are full when deciding to generate the path that goes through all said nodes. The consensus is that the fewer nodes there are, the more accurate the algorithms are in generating the shortest path possible and the less time required to do so. The more nodes available, the more complicated it is for the system to compute a possible shortest route, while taking much longer time if an insufficient algorithm is chosen. Thus, for the number of nodes less than 15, the Branch and Bound method should be preferred, as it is guaranteed to deliver the shortest path, while costing less than or equal to the Brute Force search, where every possible path is calculated. For a more complicated system of over 30 nodes, the problem becomes too complicated to solve in a reasonable amount of time for most algorithms; therefore, the most optimal algorithm in terms of balancing computing time and generating a reasonably short path is the Nearest Neighbor algorithm, in which the amount of times a path is generated is only equal to the amount of available nodes. Afterwards, the results are compared to sort out the shortest path of all generated paths. The last scenario, where the number of full nodes is between 15 to 30, can be handled with either the Space Filling Curve method or the Ant Colony Optimization method. Both can be computed in a reasonably short time for said amount of nodes, while delivering a good result but not a guaranteed shortest route.

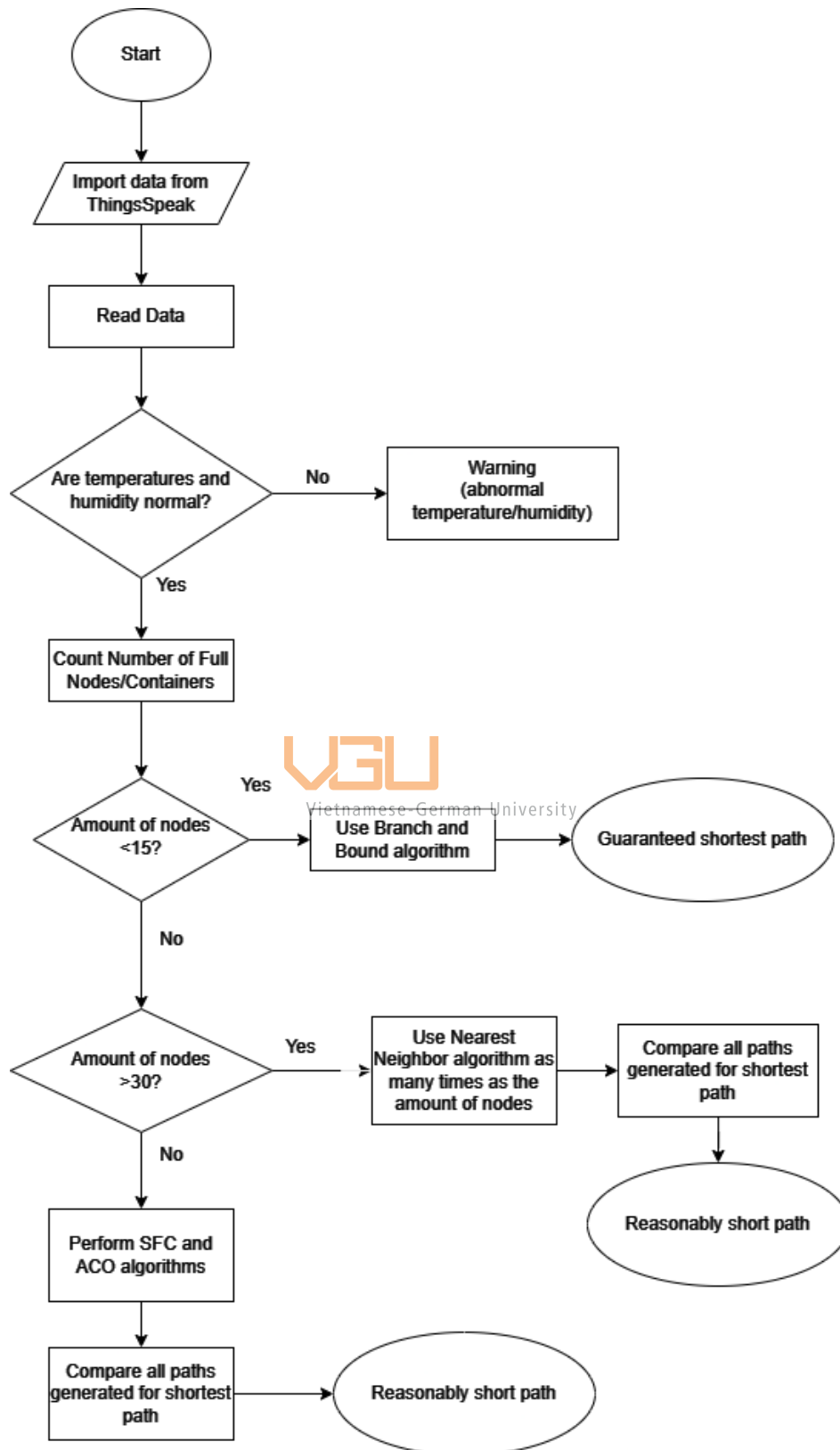


Figure 42 Path Generation Flow Chart



### 3.4 Experimental Setup

To test out the process of monitoring, transferring and uploading of data, as well as prototyping the enclosure for the monitoring units, an experimental set up is required. For this experiment, all the components must be correctly installed inside the enclosures, and a functioning code must be loaded to both the monitoring units and the upload server. The enclosures are simply 3D printed with either FDM or SLA technology for a durable prototyping unit. The code and the connection scheme for all the components can be found below:

Device	Pins	Connection	Notes
DHT11	+	5V	Common 5V power source
	-	GND	Common ground
	Out	Pin D3 on NodeMCU board	Pin 17 on Heltec LoRa board
HC-SR04	Vcc	5V	Common 5V power source
	Gnd	GND	Common ground
	Trig	Pin D2 on NodeMCU board	Pin 22 on Heltec LoRa board
	Echo	Pin D1 on NodeMCU board	Pin 23 on Heltec LoRa board

Table 5 Sensor connections

With the above connections, the sensor data can be read by the NodeMCU board and transmitted through LoRa SX1278 to the server, which is built by making the connection for the LoRa SX1278 module to the NodeMCU ESP8266:

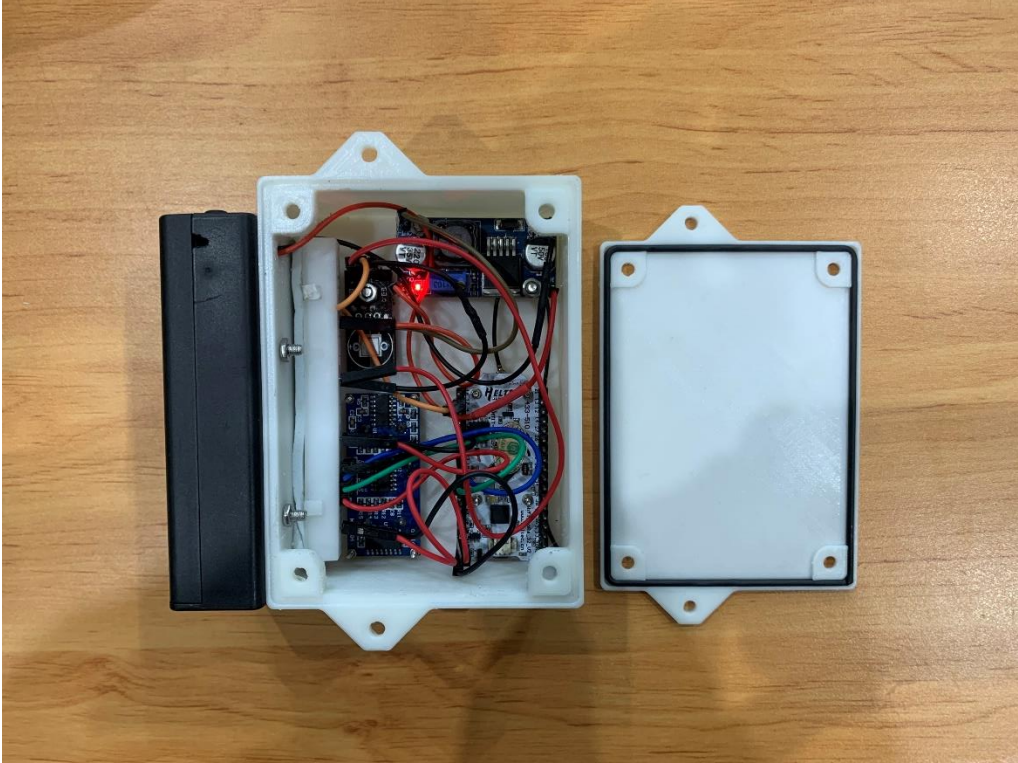
LoRa SX1278 Module	NodeMCU ESP8266
VCC	3.3V
GND	GND
NSS	Pin D8
SCK	Pin D5
MISO	Pin D6
MOSI	Pin D7
RST	Pin D0

*Table 6 Connections of LoRa module to microcontroller*

Both microcontrollers are powered by connecting their 5V and ground pins to the common 5V power source and ground pins.



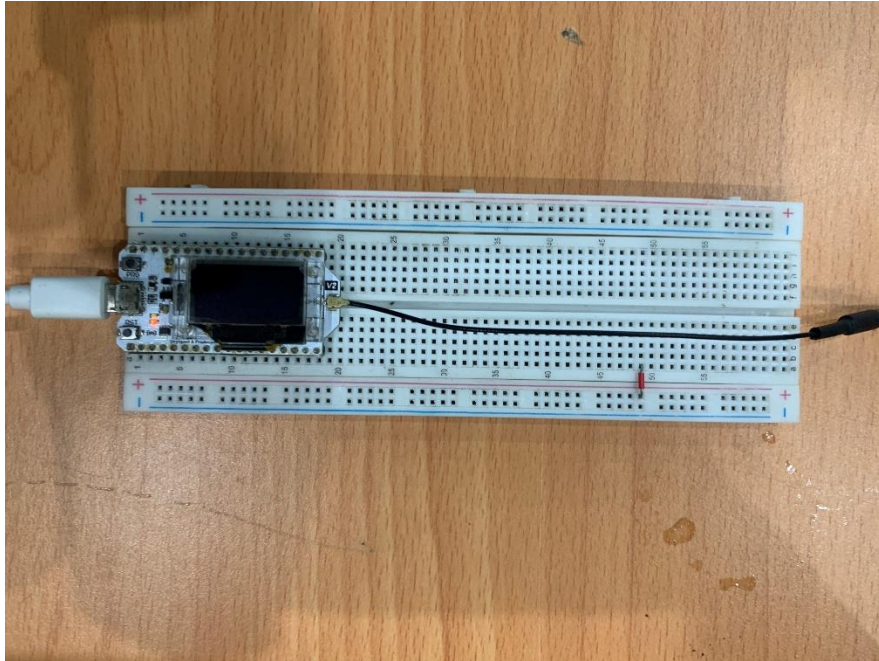
*Figure 43 The enclosure before installation*



*Figure 44 Heltec Wi-Fi LoRa enclosure fully installed*



*Figure 45 NodeMCU enclosure fully installed*



*Figure 46 Heltec Wi-Fi LoRa board as the server*



*Figure 47 Experiment Placement*

The test was carried out at the laboratory in cluster 5 of VGU Campus, with the units installed in trash bins available at cluster 1 and 2. The distance of the units from the server was around 60m away.

The programming for the monitoring units and the server can be displayed by the snippets of code below. The monitoring units' difference in programming is mainly how LoRa is initiated on each device, otherwise the code for data collection is identical between the two microcontrollers.

### Programming for LoRa Server:

- Connect server to Wi-Fi by entering ssid and password, and connect to ThingsSpeak with a generated API Key from ThingsSpeak

```
String apiKey = "0V6YO68JKTFC0Z5B";  
const char *ssid = "VGU_Student_Guest";  
const char *password = "";  
const char* server = "api.thingspeak.com";  
WiFiClient client;  
WiFi.begin(ssid, password);
```

- Initiating LoRa at the 433Mhz frequency and set syncword to the same one on the monitoring units. Listen for data through LoRa

```
Heltec.begin(true, true, true, true, 433E6);  
LoRa.setSyncWord(0xF3);  
LoRa.setSpreadingFactor(7);  
LoRa.receive();
```

- Decipher the data received through LoRa

```
while (LoRa.available()) {  
String LoRaData = LoRa.readString();  
int pos1 = LoRaData.indexOf('/');  
int pos2 = LoRaData.indexOf('&');  
int pos3 = LoRaData.indexOf('#');  
DEVICE = LoRaData.substring(0, pos1);  
temperature = LoRaData.substring(pos1 + 1, pos2);  
humidity = LoRaData.substring(pos2+1, pos3);  
fill = LoRaData.substring(pos3+1, LoRaData.length());
```

The code to decipher the message works as follows: the message is encoded in a specific way: the monitoring unit sends a string, which contains the device number and sensor data separated by special characters such as /, &, #. By determining the positions of those dividers, the program can split the received string into smaller strings that contain only the relevant data.

- Mapping data to respective variables

```
if (DEVICE == "x")
Hum(x) = String(humidity);
Temp(x) = String(temperature);
Fill(x) = String(fill);
```

The program checks for the device number of the data string, then updates the variables corresponding to the device with the new data.

- Uploading data to ThingsSpeak

```
if (client.connect(server, 80)) {
String postStr = apiKey;
postStr += "&field(x)=";
postStr += String(Temp(x));
postStr += "&field(x+1)=";
postStr += String(Humx);
postStr += "&field(x+2)=";
postStr += String(Fill(x));
client.print("POST /update HTTP/1.1\n");
client.print("Host: api.thingspeak.com\n");
client.print("Connection: close\n");
client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
client.print("Content-Type: application/x-www-form-urlencoded\n");
client.print("Content-Length: ");
client.print(postStr.length());
client.print("\n\n");
client.print(postStr);
```



### Programming for NodeMCU monitoring unit:

- Set the pins input of LoRa on NodeMCU board, initiate LoRa at the 433Mhz frequency and set syncword to the same one on the server.

```
LoRa.setPins(ss, rst, dio0);  
LoRa.begin(433E6)  
LoRa.setSyncWord(0xF3);  
LoRa.setSpreadingFactor(7);
```

- Get humidity, temperature and ultrasonic sensor data, and calculate corresponding fill level.

```
float humidity = dht.readHumidity();  
float temperature = dht.readTemperature();  
digitalWrite(trigPin, LOW);  
delayMicroseconds(2);  
digitalWrite(trigPin, HIGH);  
delayMicroseconds(10);  
digitalWrite(trigPin, LOW);  
duration = pulseIn(echoPin, HIGH);  
distance = duration * 0.034/2;  
fill = (90-distance)*100/80;
```



- Encode the data collected as a specific string and send via LoRa.

The variables are separated by special characters such as /,&,# so the message can be sent to the server in a single package, where it eventually can be split back into individual values. The message is sent as a singular string so the server can pick up the signal with ease and decipher it as explained above.

```
LoRaMessage = String(DEVICE) + "/" + String(temperature) + "&" + String(humidity) + "#" +  
String(fill);  
LoRa.beginPacket();  
LoRa.print(LoRaMessage);  
LoRa.endPacket();
```

### Programming for Heltec Wi-Fi LoRa monitoring unit:

- Initiate LoRa at the frequency 433Mhz and set syncword.

```
Heltec.begin(true, true, true , true ,433E6);  
LoRa.setSyncWord(0xF3);  
LoRa.setSpreadingFactor(7);  
LoRa.receive();
```

- Get humidity, temperature and ultrasonic sensor data, and calculate corresponding fill level.

```
float humidity = dht.readHumidity();  
float temperature = dht.readTemperature();  
digitalWrite(trigPin, LOW);  
delayMicroseconds(2);  
digitalWrite(trigPin, HIGH);  
delayMicroseconds(10);  
digitalWrite(trigPin, LOW);  
duration = pulseIn(echoPin, HIGH);  
distance = duration * 0.034/2;  
fill = (90-distance)*100/80;
```



- Encode the data collected as a specific string and send via LoRa.

The variables are separated by special characters such as /,&,# so the message can be sent to the server in a single package, where it eventually can be split back into individual values. The message is sent as a singular string so the server can pick up the signal with ease and decipher it as explained above.

```
LoRaMessage = String(DEVICE) + "/" + String(temperature) + "&" + String(humidity) + "#" +  
String(fill);  
LoRa.beginPacket();  
LoRa.print(LoRaMessage);  
LoRa.endPacket();
```



# 4. Results

## 4.1 Data transfer

The experiment above was carried out for a duration of 15 minutes, with each monitoring unit sending data continuously to the server. The results are as predicted, data flow was stable and uninterrupted by any obstacle between the buildings. The experiment was also able to test for an abnormality in the data received, which would indicate a faulty sensor or any mismatch in wiring. Below, the ultrasonic sensor for the second device was intentionally wired incorrectly to test if the system would still transfer data, instead of not responding entirely. The transmission of any abnormal values would still be beneficial compared to a non-responding unit, because the system can be diagnosed more properly. Temperature and humidity data are as expected, as they remain stable and do not fluctuate by a large margin. The fill level was tested to change according to a real world scenario where the trash level would change.



Figure 48 Experiment monitoring result

## 4.2 Power Consumption

As part of the experiment, the power consumption of each device is also measured to predict the battery life of the system.

The measured current of the board is as follows: one end of the multimeter connects to the positive output of the voltage regulator, and the other end is connected to the circuit, making the multimeter connected in series to the circuit. The displayed amperage is the current draw of the circuit at full load.

The capacity of the battery used to test is 2400mAh = 2.4Ah

Measured current with full load: 51.3mA

Power Consumption at full load:  $P = U.I = 5V \times 0.0513A = 0.2565W$

Battery life with the monitoring unit turned on at all times:

$$T = \frac{A}{I} = \frac{2.4}{0.0513} \approx 46.8 \text{ hours}$$

Where T is the time in hours, A is the amp hours (battery capacity), P is the power output/usage. With a small 2400mAh battery running a 0.2565W system continuously, it would last 46.8 hours. Therefore, with the power plan discussed above where it only turns on for 5 minutes then sleep until the next hour, battery life is improved by 1200%, which means 516.6 hours. Using a larger 7000mAh battery (2 18650 3500mAh batteries connected in series) will nearly triple the run time of the system, at 1638 hours. Shortening the span of data transfer to only 3 minutes can also lengthen the battery life by a further 66% ( $5/3 \times 100$ ), resulting in a battery life of 2719 hours, or 113 days continuously.

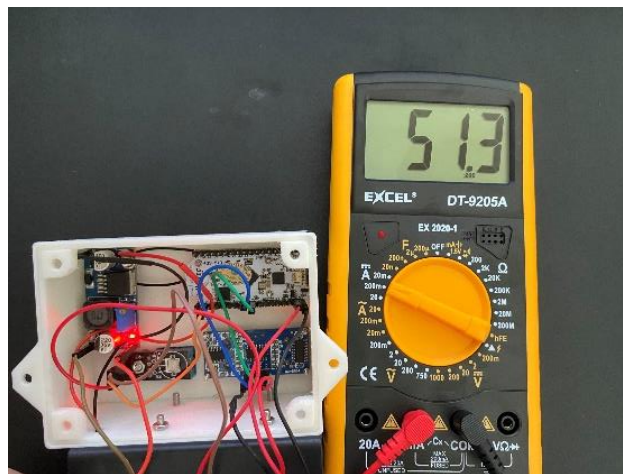


Figure 49 Measured current

### 4.3 Bill of Materials for the experiment

Below is the BoM for the experiment performed. There are several changes compared to the BoM for the designed system, notable the price point of the sensors, which was replaced by more affordable options suitable only for testing purposes.

Part number	Part name	Part description	Quantity	Price per unit	Total price
1	Heltec Wi-Fi LoRa v2	Heltec development board with LoRa	2	Sponsored	Sponsored
2	NodeMCU ESP8266 LUA CH340	Microcontroller with Wi-Fi	1	68,000 VND	68,000 VND
3	LoRa Ra02 SX1278	LoRa module	1	140,000 VND	140,000 VND
4	HC-SR04	Ultrasonic sensor	2	20,000 VND	40,000 VND
5	DHT11	Temperature and Humidity sensor	2	20,000 VND	40,000 VND
6	LM2596	Voltage regulator	2	17,000 VND	34,000 VND
7	Wires		1	20,000 VND	20,000 VND
8	Breadboard		1	11,000 VND	11,000 VND
9	Screws	Multiple sizes	1	40,000 VND	40,000 VND

Table 7 Bill of Materials for experimental set up

## 5. Discussion and Conclusions

Whilst the results of the experiment set up at VGU campus is quite fruitful in terms of proving the capability of the data transfer of the system, it only reflects a portion of the smart waste management system proposed in this thesis. Given the lack of major funding and a mild time constraint, the programming for an application to test out a large number of devices (10, 15, 20) is unattainable at the moment of writing. However, the guidelines and strategies presented in this thesis should still work in practice, since the only limitations this thesis faced was the amount of available hardware and the time to implement the algorithms. Otherwise, all the programming and installation of the components are scalable and should operate normally with larger numbers of devices.

In conclusion, this thesis has presented a probable smart waste management system that covers from the user end, which is the design of the trash containers, to the data monitoring and processing of the management end. It does not discuss the processing and further categorization of wastes, as the current waste handling situation present in Vietnam proves unnecessary for any single institution to categorize trash into their most correct type, which is done by a separate professional facility. The system at the moment features a design that helps users recognize the most basic two types of trash: organic and inorganic. The monitoring system is tested to work flawlessly with its power consumption is as expected. The algorithm to generate the shortest waste collection route can be further developed given a suitable amount of time.

The trial application of this thesis is as previously stated, to be purely experimental, to test the capability of data transfer of the monitoring units and the server. Thus, there are plenty of room for adjustments, advancements, and expansion, such as optimizing the timing of the packages to further reduce power consumption or customizing a development board with the barebones microcontrollers and the LoRa module, alongside the sensors, to create a much more space-saving enclosure and therefore reduced costs. The possibilities to improve this system is vast, and much needed if commercial distribution is the final goal. However, certain core concepts and guidelines must be kept in tact, in order to maintain operability and comprehension for the system. Such guidelines are:

- The system must feature a trash bin that is designed to split trash into two categories.

- The system must operate using a technology different from Wi-Fi to ensure good coverage for a large area (LoRa, radio, Zigbee, etc.)
- If an IoT platform is chosen to be the upload destination of collected data, it must be capable of exporting said data to a suitable format for the application that generates the collection route to read.
- The battery life should last for a long time.

Remark: it is essential that anyone attempting to produce a smart waste management system accounts for real life scenarios to design the system accordingly. The system designed here takes into account the situation present at the Vietnamese – German University and performs its experiments there. Any alteration and improvement of this system should result in a better functioning system when applied to the current scene at VGU.



## 6. References

- [1] World Bank, "The World Bank Annual Report 2014," Washington, DC., 2014.
- [2] S. Kaza, L. C. Yao, P. Bhada-Tata and F. Van Woerden, What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050, Washington, DC: World Bank, 2018.
- [3] D. Knapp and M. L. Deventer, "Glass Root Recycling Network," 2002. [Online]. Available: [https://archive.grrn.org/zerowaste/twelve\\_categories.html](https://archive.grrn.org/zerowaste/twelve_categories.html).
- [4] J. Yu, "Traveling salesman problem," 2014. [Online]. Available: [https://optimization.cbe.cornell.edu/index.php?title=Traveling\\_salesman\\_problem](https://optimization.cbe.cornell.edu/index.php?title=Traveling_salesman_problem).
- [5] Science Daily, "Science Daily," [Online]. Available: [https://www.sciencedaily.com/terms/waste\\_management.htm](https://www.sciencedaily.com/terms/waste_management.htm).
- [6] I. H. Sarker, "Data Science and Analytics: An Overview from Data-Driven Smart Computing, Decision-Making and Applications Perspective," *SN Computer Science*, 2021. Vietnamese-German University
- [7] S. Cheema and A. Hannan, "Smart Waste Management and Classification Systems Using Cutting Edge Approach," *Sustainability*, 2022.
- [8] M. Dupré, "Categorization and Sorting for Waste Management," *International Journal of Waste Resources*, 2016.
- [9] "NodeMcu ESP8266 V3 Lua CH340 Wifi Development Board," Techtonics, [Online]. Available: <https://www.techtonics.in/nodemcu-esp8266-v3-lua-ch340-wifi-development-board>.
- [10] Heltec Automation, "WiFi LoRa 32 (V2.1) Phaseout," Heltec Automation, [Online]. Available: <https://heltec.org/project/wifi-lora-32/>.
- [11] Semtech, "Product Details SX1278," Semtech, [Online]. Available: <https://www.semtech.com/products/wireless-rf/lora-connect/sx1278>.
- [12] S. Robotics, "IOE-SR05 Ultrasonic Distance Sensor Module [3V-5.5V] [0-200cm]"

- [13] E. Fahad, "DHT21 AM2301 Temperature & Humidity Sensor".
- [14] Arduino, "Arduino Nano," Arduino, [Online]. Available:  
<https://store.arduino.cc/products/arduino-nano>.
- [15] ICdayroi, "nextion 7 inch display NX8048T070-011".
- [16] HShop.vn, "mmWave Human Presence Sensor Rd-03".
- [17] V.O.V, "Every year Vietnam generate 1.8 million tonnes of plastic wastes," 2023.
- [18] A. Tripathy, "Travelling Salesman Problem (Basics + Brute force approach)".
- [19] GeeksforGeeks, "Traveling Salesman Problem using Branch And Bound".
- [20] G. Kizilates-Evin and F. Nuriyeva, "On the Nearest Neighbor Algorithms for the Traveling Salesman Problem," *Advances in Intelligent Systems and Computing*, 2013.
- [21] Y. Hsieh and P.-S. You, "A New Space-Filling Curve Based Method for the Traveling Salesman Problems," *Applied Mathematics & Information Sciences*, 2012.
- [22] M. Dorigo, M. Birattari and T. Stützle, "Ant Colony Optimization," 2006.
- [23] B. P. Silalahi, N. Fathiah and P. T. Supriyo, "Use of Ant Colony Optimization Algorithm for Determining Traveling Salesman Problem Routes," *Jurnal Matematika MANTIK*, 2019.
- [24] Josephbakulikira, "Traveling Salesman Algorithm".
- [25] AndrewB330, "TSP-Solver".
- [26] Y. Glouche and P. Couderc, "A smart waste management with self-describing objects," June 2013.
- [27] EasyCalculation, "Traveling Salesman Problem Calculator".