دانشکده مهندسی برق

گروه کنترل

Final Project Report – Internet of Things 2023

Temperature Anomaly Detection in IoT Devices Using Machine Learning

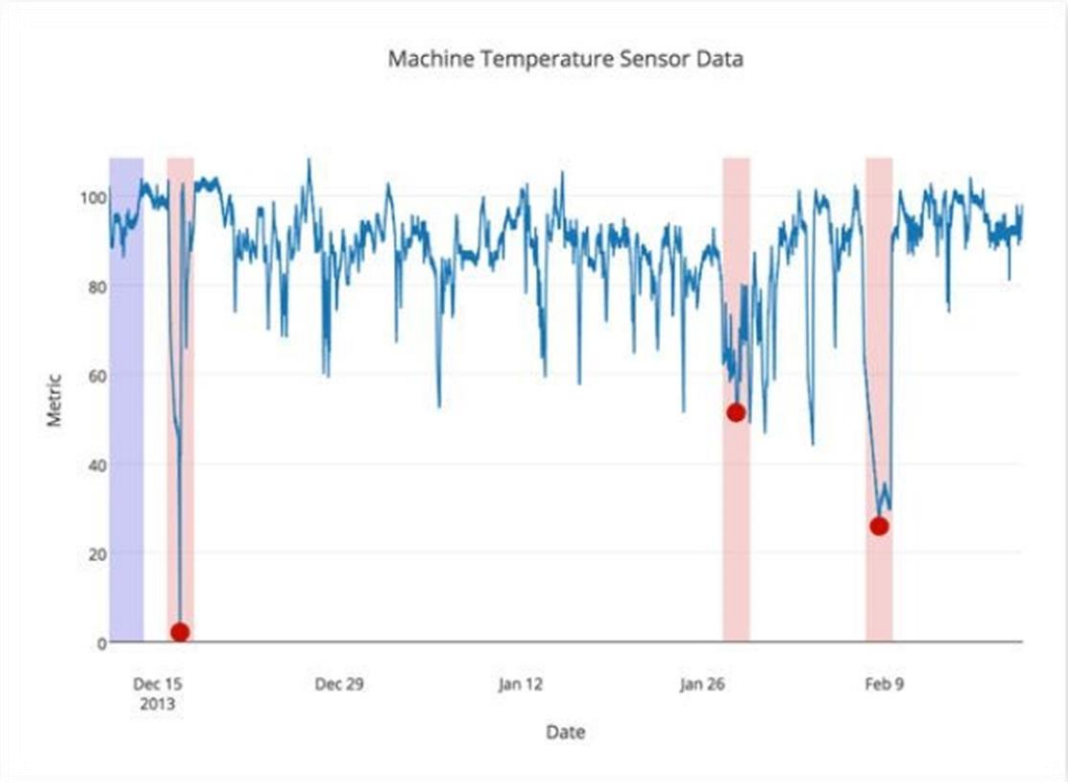
Armin Attarzadeh

Arshia Goshtasbi

Dr. Reza Vahidnia

Contents

Part Zero: Introduction and Definition of Purpose



Introduction:

In today's world, where the Internet of Things (IoT) has become a key technology, analyzing data from these devices and extracting useful insights is crucial. One of the most important applications in the industry is detecting anomalies in sensor data. In this project, we will focus on identifying anomalies in temperature data using machine learning methods.

Objectives:

The main goal of this project is to develop an anomaly detection system for temperature data collected from sensors in IoT devices. This system will be able to automatically and intelligently identify anomalies, which may be caused by factors such as sensor failure, environmental issues, or even malicious attacks on the network.

The specific objectives of this project include:

- Understanding IoT technologies and related concepts
- Presenting machine learning methods for detecting anomalies in temperature data
- Implementing a practical and usable system for anomaly detection in sensor data
- Providing an encrypted communication system alongside a database implementation

Applications of Temperature Anomaly Detection:

Detecting fluctuations or anomalies in temperature data is critical and has applications in various industries and areas where security and efficiency are important:

Equipment Failure Prevention: In industrial settings like factories, temperature is a vital parameter. Detecting fluctuations early can provide warnings of equipment failures or malfunctions, helping to prevent further damage.

Monitoring: In environments that require strict temperature control, such as server rooms or warehouses, detecting anomalies helps in more effective monitoring and management of these spaces.

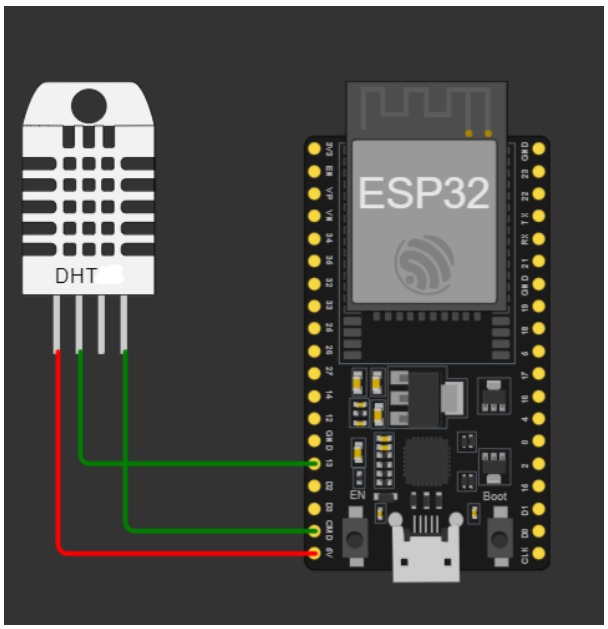
Smart Farming: In agriculture, monitoring temperature is key to crop growth and harvesting. Detecting sudden changes can prevent issues like drought or irrigation system failures.

Improving Health and Safety: In settings like hospitals, kitchens, and public spaces, monitoring temperature changes can help improve health and safety, preventing issues such as the spread of diseases.

In summary, using machine learning methods and anomaly detection algorithms to monitor temperature data can enhance performance and security across a wide range of industrial and non-industrial systems.

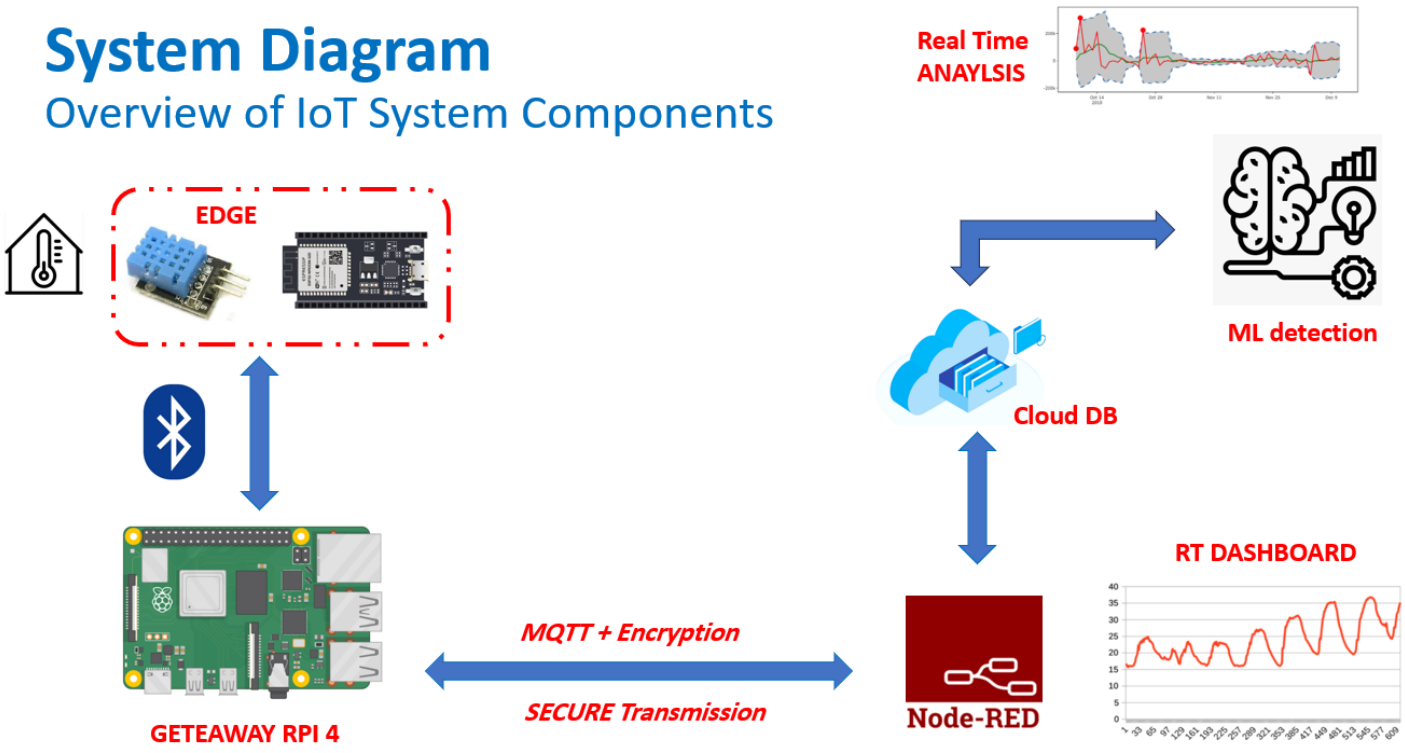
Part One: IoT Hardware

In the EDGE layer, an ESP۳۲ and a DHT۱۱ sensor are used to send data to the Raspberry Pi. An application in the Arduino IDE handles this task. One part of the program reads the data from the sensor, and the other part sends it via Bluetooth. The NRF Connect app is also used to check this process. Circuit schematic related to ESP۳۲:



The code for this section is available in the sendDHTBLE.ino file.

For the general system diagram block, we have:



Part Two: IoT Communication

In this project, two types of communication have been used:

1-Data exchange between the ESP32 and the Raspberry Pi is accomplished using Bluetooth. For this purpose, two programs are employed—one written in C and the other in Python. In the Python part, the Bluepy library is used to facilitate Bluetooth communication.

2-Data exchange between the Raspberry Pi and the Cloud is managed through the MQTT protocol. Prior to being sent to the Cloud, the data is encrypted using the RC4 algorithm to ensure security.

The Python code for this section is divided into three distinct parts:

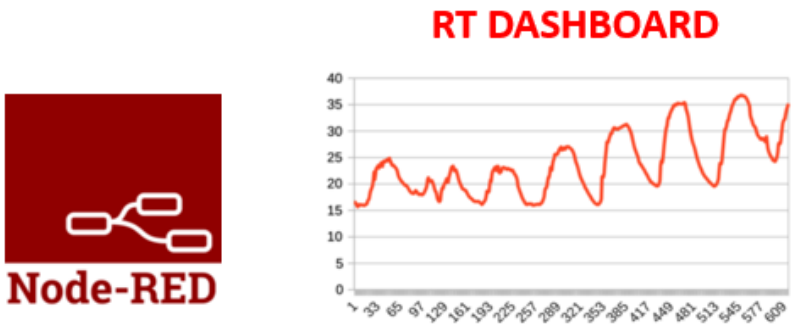
1-Retrieving data from Bluetooth, which is handled by the Bluepy library.

2-Encrypting the data using the RC4 algorithm, which is implemented through Python functions.

3-Sending the encrypted data to the Cloud, utilizing the paho-mqtt library for MQTT communication.

The complete program code for these operations can be found in the file named sendToCloud.py.

Part Three: Data Display and Dashboard



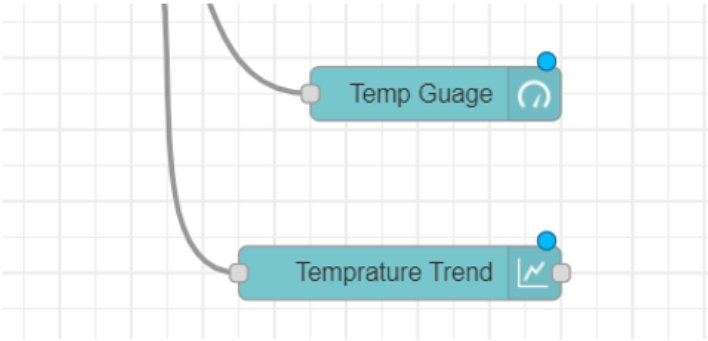
For the data visualization discussion, the Node-RED dashboard was used to create a user interface for displaying temperature data. After the data is collected from the sensor and transmitted securely through cloud servers, it is decrypted in the Node-RED programming environment. The numerical temperature data is then displayed in real time on a time graph.

A gauge was used to represent the current temperature, and a chart was used to show the temporal trend of the temperature.

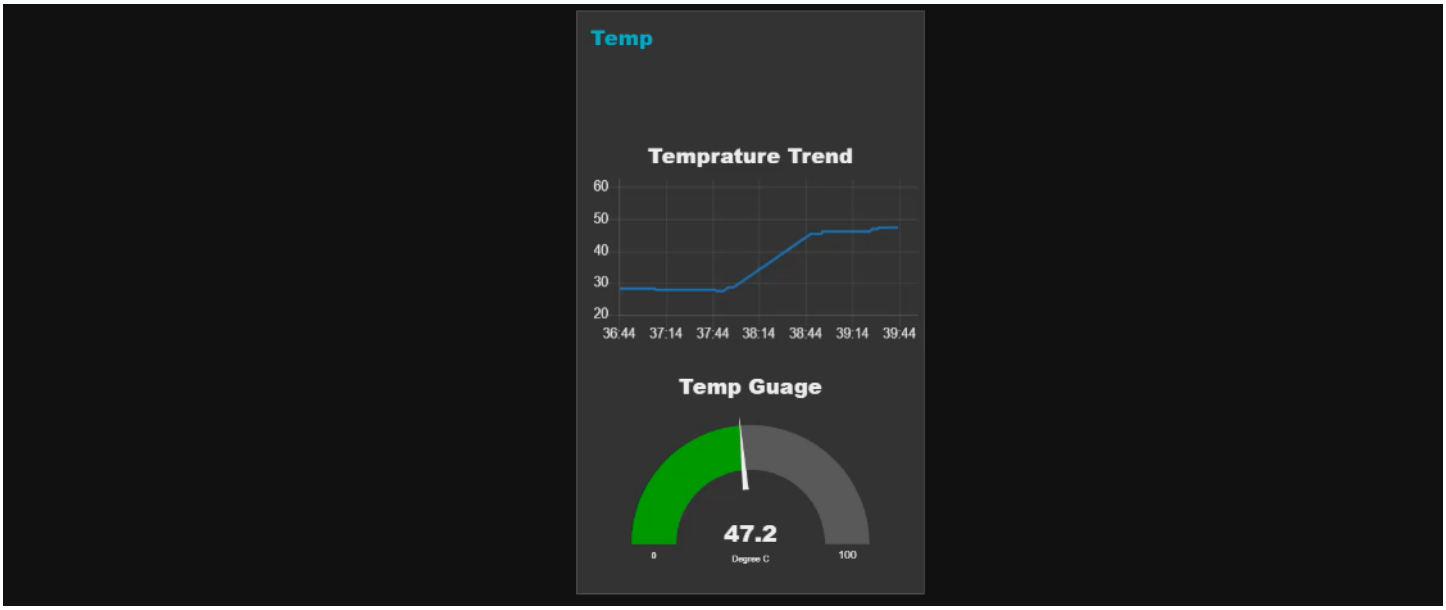
To set up this user interface, the following library needs to be installed in the Node-RED environment:

node-red-dashboard 3.6.2

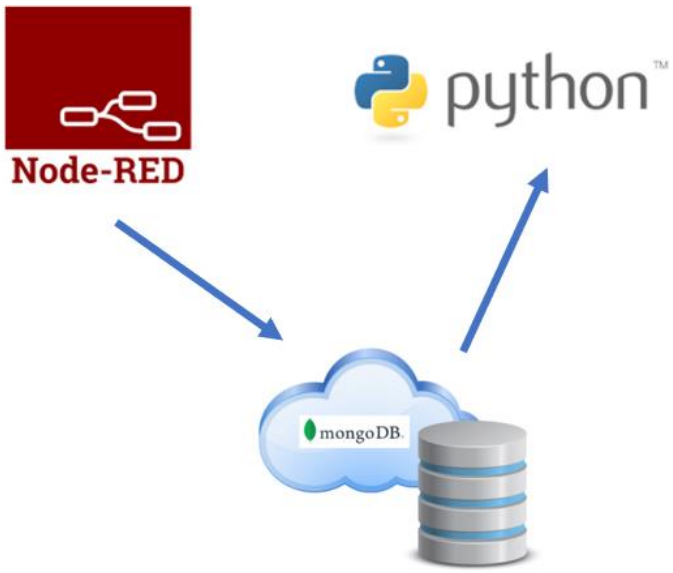
After installation, the following blocks were used:



A demonstration of the data visualization system in the Node-RED dashboard



Part Four: Database and Data Storage



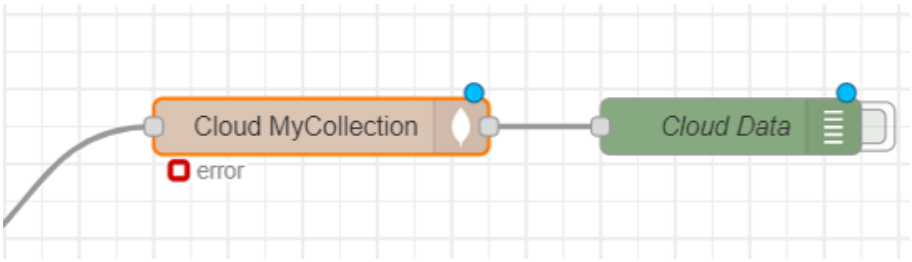
Along with the visualization component, the data needs to be properly stored in the cloud database after decryption. In this project, the free MongoDB cloud service was utilized. In the Node-RED environment, after installing the relevant library, data can be stored in the cloud as JSON. Once a collection is created on the MongoDB site, the data can be stored in it or read from it.

After storing the data, it is accessed in the Google Colab environment using Python for use in the machine learning algorithm.

To store the temperature values in the database in the Node-RED environment , install the following library:

node-red-contrib-mongodb-aleph 0.3.0

Data can be stored with the MongoDB out block.



In the block, the connection to the server, the password and the collection name must be created and used from the MongoDB website . The following image shows the information stored in the database:

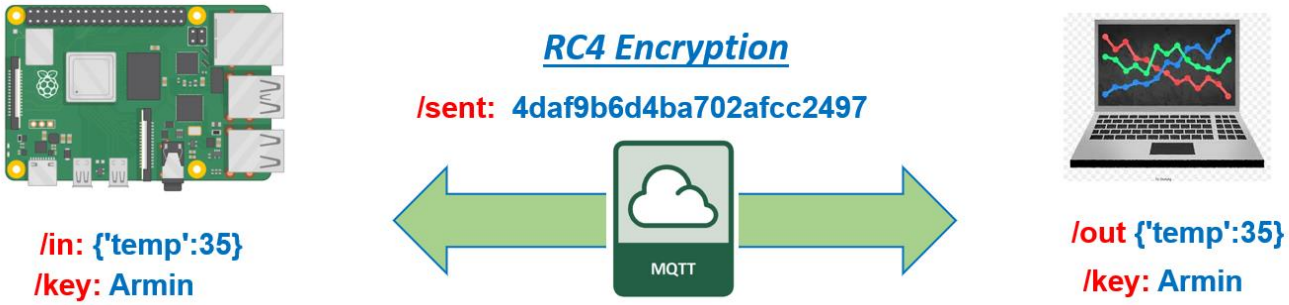
QUERY RESULTS: 1-20 OF MANY

```
_id: ObjectId('65b3d3ad01da125fe019f048')
payload: 26.3
_msgid: "c8ea51d0d5658175"
```

```
_id: ObjectId('65b3d3ae01da125fe019f049')
payload: 26.3
```

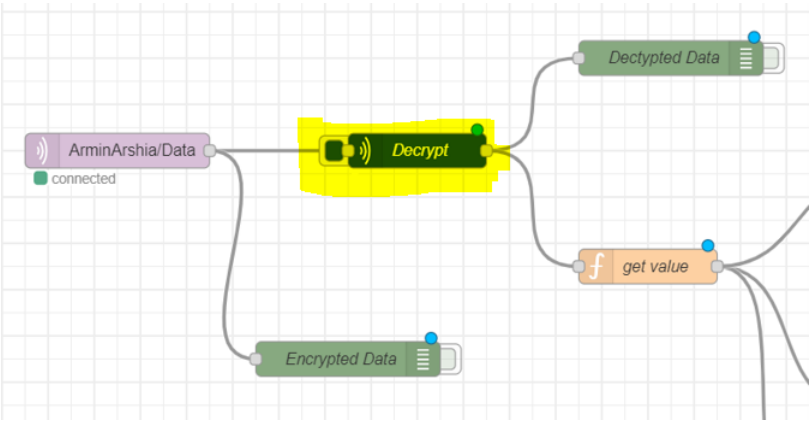
In the data analysis phase with machine learning, this database is used to analyze and diagnose anomalies

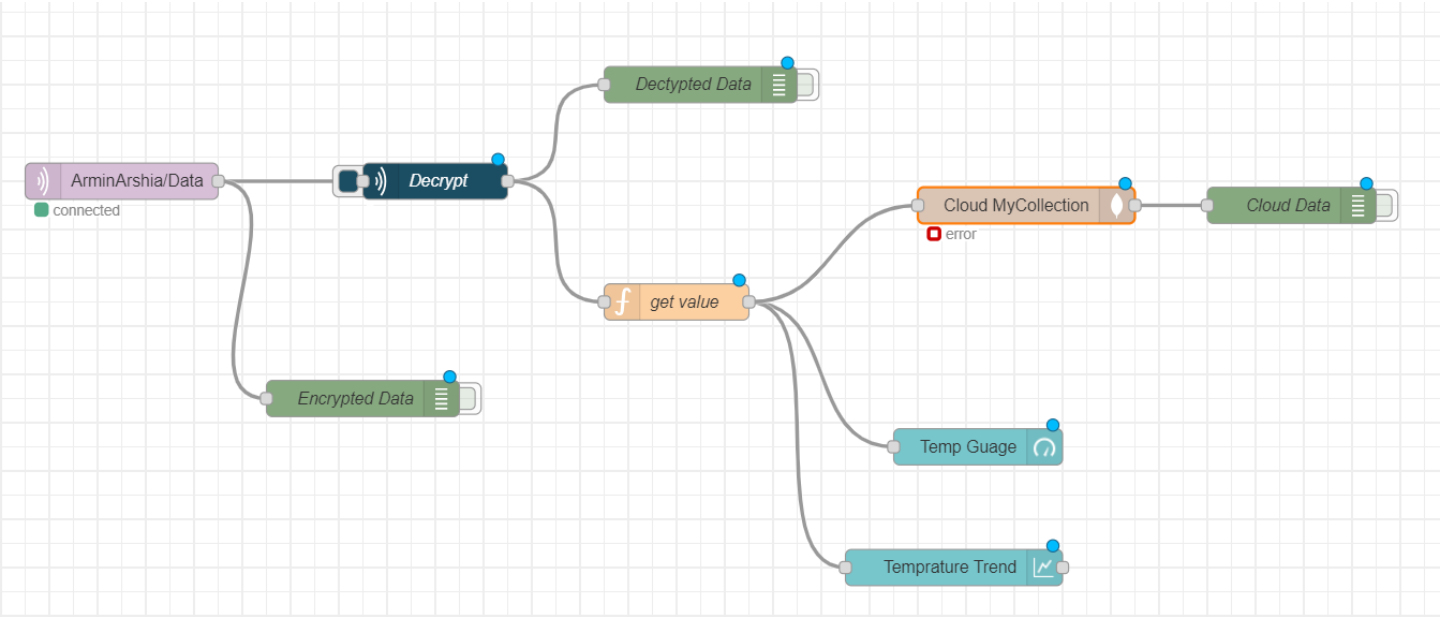
Part Five: Data Encryption and Decryption



To enhance the security and reliability of the system, data is encrypted before connecting to the MQTT servers and decrypted on the receiving side. Several algorithms were tested and evaluated, and the RC4 decryption algorithm was ultimately chosen for this project. Both the origin and destination sides can access the data by knowing the password or key.

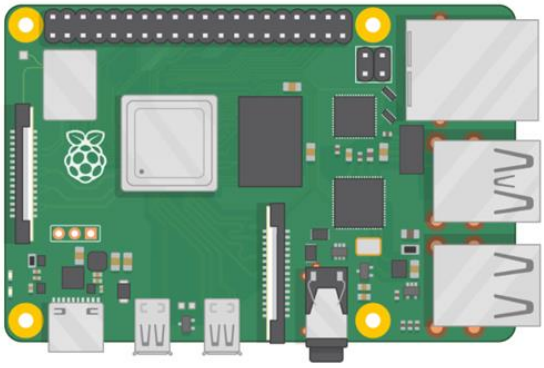
There are two codes related to the RC4 algorithm: Encrypt.py and Decrypt.py. Encryption is performed on the Gateway side, while decryption and usage of the data occur in Node-RED. An image illustrating the decryption process is shown below:





The Node-RED blocks for this file are stored under the title `final nodered`.

Part Six: Raspberry Pi as Gateway



In the first concession section, a Raspberry Pi is used as a gateway. This setup allows for the connection of multiple ESP۳۲ devices to the project simultaneously. The Raspberry Pi's tasks in this section are as follows:

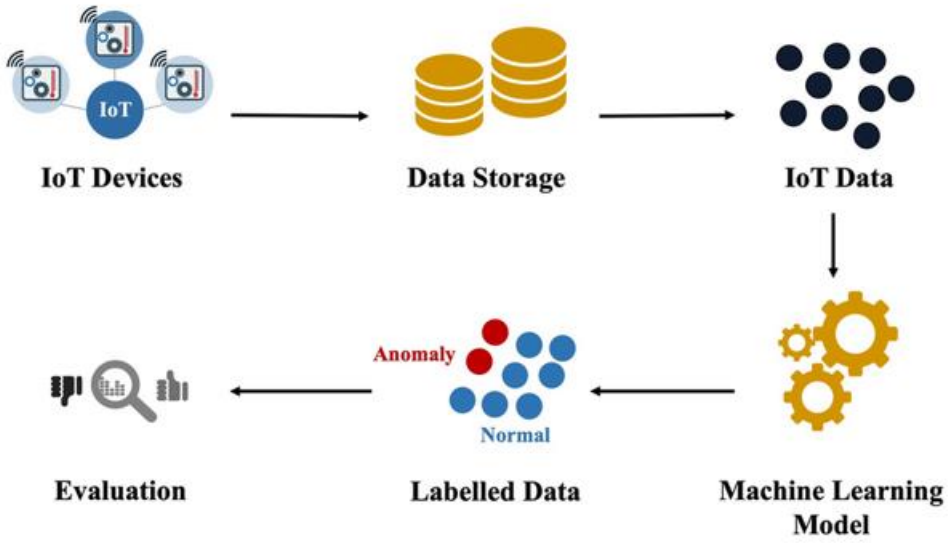
1-Retrieve data from Bluetooth (using the Bluepy library)

2-Encrypt the data with RC۴ (implemented using Python functions)

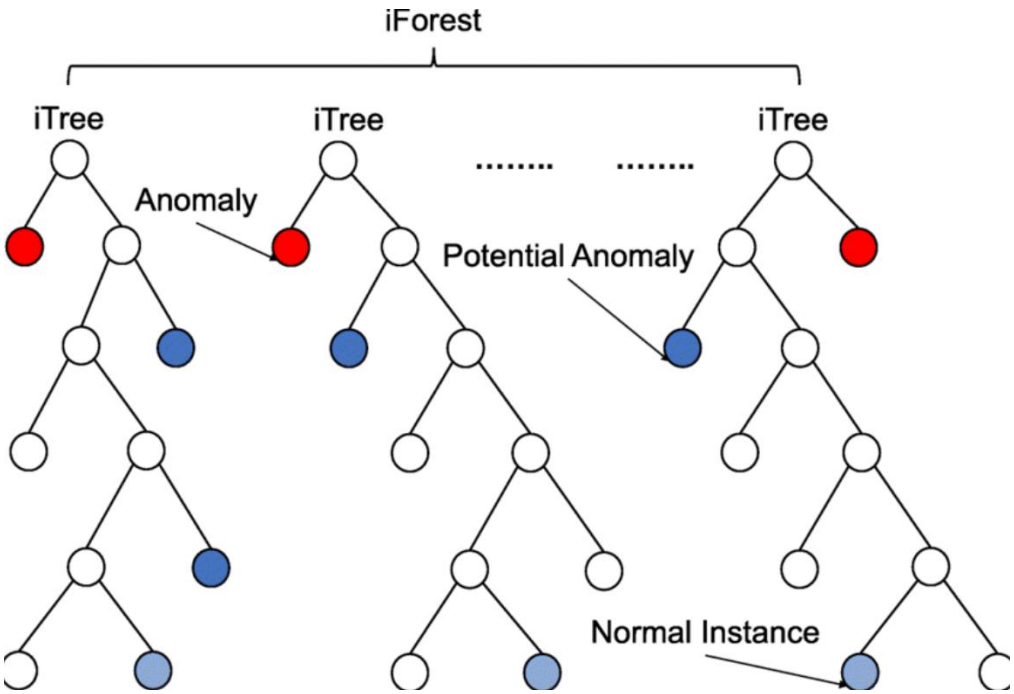
3-Send the data to the cloud (using the paho-mqtt library)

The program code for this section is available in the `sendToCloud.py` file.

Part Eight: Machine Learning and Data Analytics



General Data Processing Trends for Machine Learning



As the second scoring part of the project, machine learning algorithms were employed to analyze the data. The goal is to detect anomalies in the time series of temperature data. One of the most commonly used algorithms for this purpose is Isolation Forest, which classifies data to differentiate between anomalous and normal data. This algorithm is unsupervised learning and does not require labeled data.

The implementation method operates in pseudo-real time: every 10 seconds, data is read from the database, and the machine learning algorithm performs anomaly detection in the subsequent step.

```
while True:
    # Read temperature data
temperature_data = read_temperature_data()

    # Perform anomaly detection and visualization
perform_anomaly_detection(temperature_data)

    # Wait for 10 seconds
    time.sleep(10)
```

The relevant codes are typed in the Google Colab environment and can be seen in the folder that accompanies this report.The end result is summarized by analyzing about 7000 data as follows: