A report on
Final Assignment
of
IoT Systems

submitted in partial satisfaction of the requirements for the degree of

Bachelor of Engineering
in
Electrical and Automation

by

Acharya Hari

Table of content

1. Functional Definition

1.1 System Functions and Interfaces

- A short description about exercise

In this exercise I have learned about Raspberry Pi minicomputer, Arduino boards and practical usage of those.

I wanted to connect one Arduino board to a laptop via its serial port, and I also needed to connect the Raspberry Pi to that laptop via network cable and another Arduino Uno WiFi card to the Raspberry Pi.

Then I want to open inbuilt Node-RED system of Raspberry pi and read the sensor readings of Raspberry Pi and those two Arduino board sensors via Node-RED.

1.2 Necessary Devices

Here is the list of all hardware components that I used

1. Raspberry Pi 3 minicomputer

2. Arduino Uno WiFi Rev.2 microcontroller board

3. Arduino Uno microcontroller board

4. Sensors (DHT11, DHT 22)

5. Cables and connection components (Two USB-to-Serial Cables, Ethernet Cable, Raspberry Pi power supply)

1.3 Followed mechanism.

The connection steps those I followed, are summarized below,

1.3.1. Arduino to Laptop Serial Connection

Connected the USB end of USB-to-Serial Cable to laptop and connected the other end to the Arduino's serial port.

Opened Arduino IDE on the laptop and compile the necessary modifications (DHT Pin id and Sensor type for this exercise) to MQTT code which provided by teacher on Moodle and upload the code to Arduino board.

Then opened the Serial monitor of Arduino IDE and checked that everything was working fine.

1.3.2. Raspberry Pi Network Connection

Plugged in Raspberry Pi with its power adapter

Connected Ethernet cable: one end to Raspberry Pi's Ethernet port and other end to laptop's Ethernet port or network switch.

Then configured network settings:

Opened the PuTTY Software from the laptop and entered static IP:169.254.209.214

Then Username was ‘pi’ and password is ‘raspberry’ for make the connection

After that opened the Node-RED interface of Raspberry Pi by using this link http://169.254.209.214:1880/

1.3.3. Arduino Uno WiFi to Raspberry Pi Connection

Connected the USB end of USB-to-Serial Cable to Raspberry Pi

Connected the other end to the Arduino's serial port.

Made a ‘MQTT in’ Node from Node-Red and connected the Arduino with HAMK server.

1.4 Functional Requirements Specification

Here is a brief summery which specifies the functional requirements i have identified for establishing connectivity between an Arduino board, Raspberry Pi, and Arduino Uno WiFi card.

1.4.1 System Overview

The system was established three primary connectivity paths:

- Arduino board to laptop via serial port

Hardware Requirements
- Arduino board (standard model)
- USB-to-Serial cable or USB cable compatible with Arduino
- Laptop with available USB port

Connectivity Specifications
- Connection Type: USB Serial (UART)
- Baud Rate: 9600 bps (default, configurable)
- Data Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: None

Software Requirements
- Arduino IDE installed on laptop
- Appropriate USB drivers for Arduino board
- Serial port communication library

- Raspberry Pi to laptop via network cable

Hardware Requirements
- Raspberry Pi (with Ethernet port)
- Standard Ethernet network cable
- Laptop with Ethernet port or USB-to-Ethernet adapter

Network Configuration

- Connection Type: Wired Ethernet
- IP Configuration:
    - Static IP preferred
    - DHCP fallback option
- Network Protocol: TCP/IP

Software Requirements

- Raspberry Pi OS (Raspbian)
- SSH enabled for remote access (PuTTY)

- Arduino Uno WiFi card to Raspberry Pi

Hardware Requirements

- Arduino Uno WiFi card
- USB-to-Serial cable or USB cable compatible with Arduino

Wireless Connectivity Specifications

- Wireless Standard: IEEE 802.11 b/g/n
- Security Protocols:
    - WPA2
    - WPA3
- Authentication Methods:
    - WPA-PSK
    - Enterprise authentication

1.5 Connections (I/O List)

Here is a detailed table of connections

Component	Source device	Destination device	Pin/port numbers	Signal type (digital/analog)	Voltage levels	Communication protocol
DHT11	Arduino 1	Laptop		Digital		
DHT22	Arduino 2	Raspberry Pi		Digital		
Sense HAT	Raspberry Pi	Laptop		Digital		

2. System Description

2.1 Program Components

The main process of reading sensor data and make a graphical output based on Node-RED programme which was installed in Raspberry Pi.

In this exercise i have identified the core functionality of Node-RED is an open-source flow-based programming tool designed for connecting hardware devices, APIs, and online services in innovative ways.

It has Primary System Functions; I have identified some of those and summerized below

- Visual programming environment
- Event-driven flow creation
- Low-code integration platform
- Rapid application development
- IoT and middleware integration

Node-Red has key system characteristics, It provides Flow-Based Programming with graphical programming interface. In this interface i found some of characteristics. They can describe as nodes represent individual functions, connections between nodes define data flow, drag-and-drop node configuration and real-time editing and deployment.

I also explore about the key concepts in Node-RED and summarized them as follows:

1. Node

- Basic building block of a flow
- Triggered by messages or external events
- Can have one input port and multiple output ports
- Processes messages and passes them to subsequent nodes

2. Configuration Node

- Special node holding reusable configuration
- Shared across multiple nodes in a flow
- Not visible in main workspace
- Accessible through Configuration nodes sidebar
- Example: MQTT broker connection settings

3. Flow

- Represented as a tab in the editor workspace
- Organizes and connects nodes
- Can contain multiple sets of connected nodes

4. Context

Three types of context for storing shared information:

- Node context: Visible only to the node that set the value
- Flow context: Visible to all nodes in the same flow
- Global context: Visible to all nodes

Storage options:

- Default: In-memory (values lost on restart)
- Optional: File-system based for persistence
- Supports alternative storage plugins

5. Message

- Plain JavaScript objects passed between nodes
- Can have any properties
- Conventionally includes a 'payload' property with key information

6. Subflow

- Collection of nodes collapsed into a single node
- Reduces visual complexity
- Creates reusable components

7. Wire

- Connects nodes
- Represents message passage through the flow

8. Palette

- Located on the left of the editor
- Lists available nodes
- Can be extended by installing additional nodes

9. Workspace

- Main area for developing flows
- Allows dragging and connecting nodes
- Tabbed interface for multiple flows

10. Sidebar
- Contains panels with tools and information
- Provides node details, debug messages, and configuration views

Node-Red has Node Types, they can identified as Input Nodes, Processing Nodes and Output Nodes.

Here is a brief explanation on Node-RED Core Nodes that I have used and their typical uses:

1. Inject Node
   - Purpose: Manually or automatically triggers a message at specified intervals
   - Use in this Project: Sending periodic commands or messages to get data from https://openweathermap.org/api

   - General Uses:
     - Simulating data inputs
     - Triggering flows at regular time intervals
     - Testing and debugging flows
     - Sending periodic commands or messages

2. Debug Node
   - Purpose: Outputs messages to the debug sidebar for troubleshooting
   - Use in this Project: checking outputs for troubleshooting when making the flow

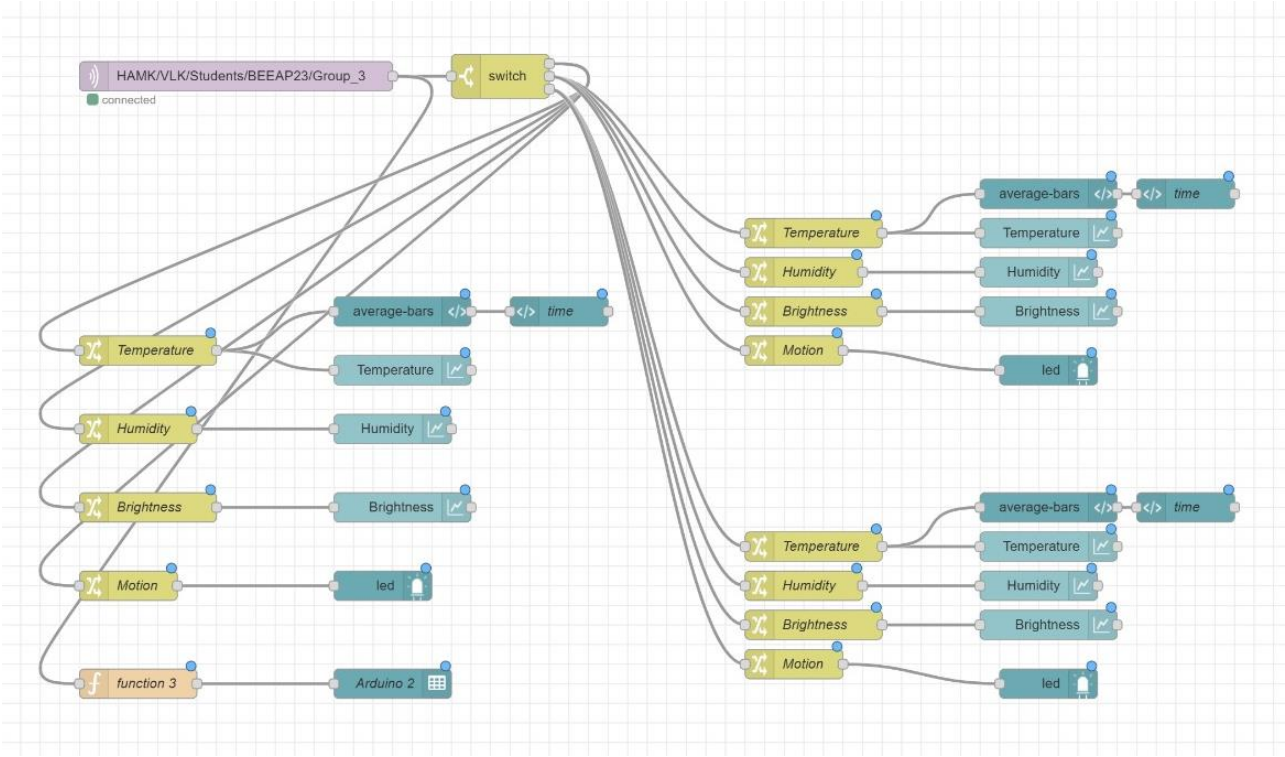
   - General Uses:
     - Inspecting message contents during flow execution
     - Verifying data transformation
     - Checking message properties
     - Diagnosing issues in complex flows

4. Change Node
   - Purpose: Modifies message properties, adds, deletes, or changes values
   - Use in this Project:

   - General Uses:
     - Data transformation
     - Setting default values
     - Manipulating message context
     - Preparing data for subsequent nodes

5. Function Node
   - Purpose: Allows custom JavaScript code to process and modify messages
   - Use in this Project:

   - General Uses:
     - Complex data manipulations
     - Custom logic implementation
     - Advanced message transformations
     - Performing calculations
     - Writing custom processing scripts



6. MQTT In Node
   - Purpose: Subscribes to MQTT topics and receives messages
   - Use in this Project: Getting Sensor readings from HAMK Server

- General Uses:

   - IoT device communication
   - Receiving sensor data
   - Home automation
   - Monitoring distributed systems
   - Collecting data from MQTT brokers

3. Switch Node
- Purpose: Routes messages based on specified conditions
- Use in this Project: Filter and make out selected topics

- General Uses:

- Conditional message routing
- Filtering messages
- Implementing logic branches
- Directing flow based on message properties

8. Timestamp Node
- Purpose: Adds timestamp information to messages
- Use in this Project: Make an outputs for HAMK server with same time intervals

- General Uses:

- Logging
- Tracking message creation time
- Performance monitoring
- Adding temporal metadata to messages

11. JSON Node
- Purpose: Parses or stringiest JSON data
- Use in this Project: Make the output message according to given format

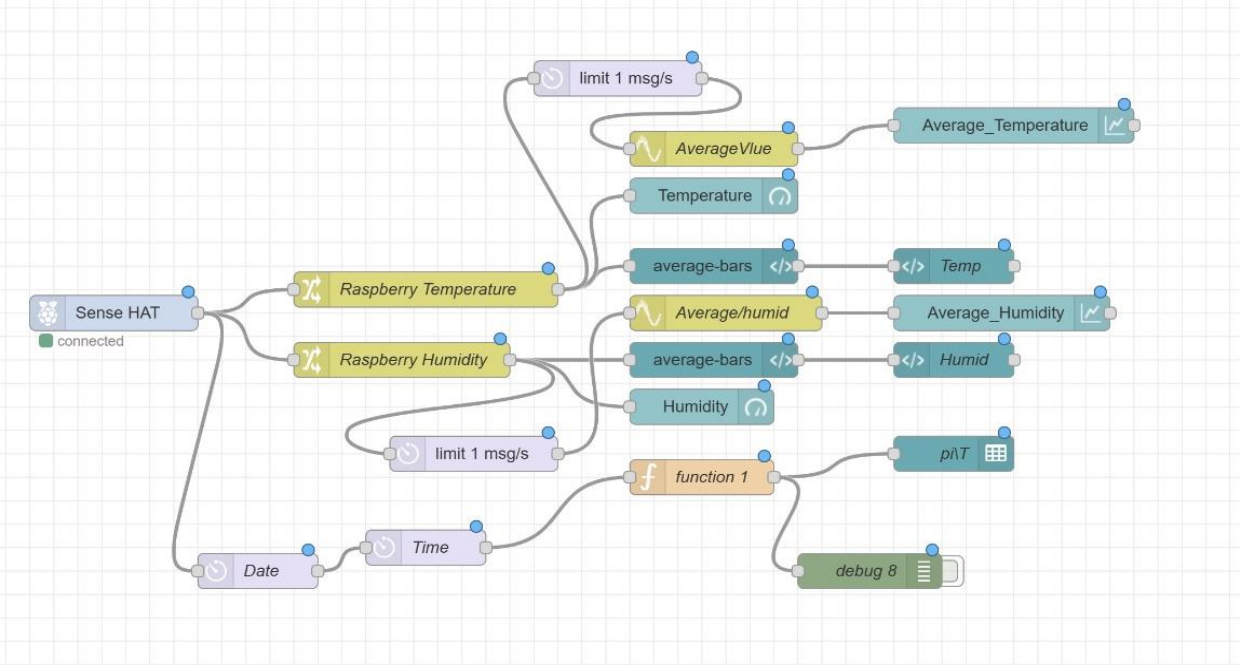
- General Uses:

- Converting between JSON and JavaScript objects
- Parsing API responses
- Preparing data for transmission
- Working with structured data formats

7. MQTT Out Node
  - Purpose: Publishes messages to MQTT topics
  - Use in this Project: Publish sensor reading values to HAMK server

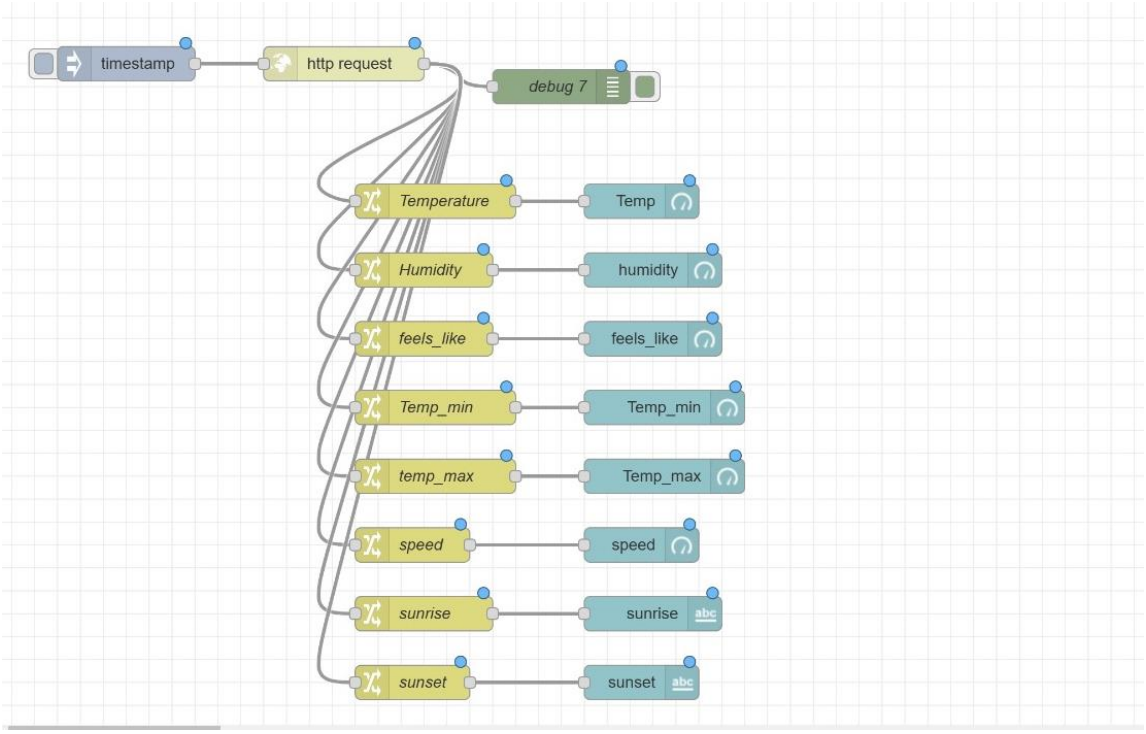
- General Uses:

  - Sending commands to IoT devices
  - Triggering actions in remote systems
  - Sharing data across different platforms
  - IoT communication and control



9. Sense HAT Node
  - Purpose: Interacts with Raspberry Pi Sense HAT hardware
  - Use in this Project: Make an output with temperature and humidity of Raspberry Pi

  - General Uses:
    - Reading environmental sensors (temperature, humidity, pressure)
    - Accessing accelerometer and gyroscope data
    - Controlling LED matrix display
    - IoT and educational projects with Raspberry Pi



10. HTTP Request Node
- Purpose: Sends HTTP/HTTPS requests (GET, POST, PUT, DELETE)
- Use in this Project: Getting data from https://openweathermap.org/api
- General Uses:
  - Fetching data from web APIs
  - Sending data to web services
  - Integrating with external web applications
  - Retrieving information from web resources

Each of these nodes plays a crucial role in creating flexible and powerful flows in Node-RED, enabling complex integrations, data processing, and automation scenarios.

Each node includes documentation accessible through the Info sidebar tab, it was easy to understand and configure their functionality to me.

2.2 Communication Protocols

2.2.1 There are several communication methods to make Raspberry Pi and Arduino Communication. In this exercise I mainly used USB communication

Characteristics of Direct USB connection:

- Plug-and-play connectivity
- No additional hardware required
- Supports high-speed data transfer
- Easy to implement

2.2.2 USB Communication Implementation

- Uses USB-to-Serial interface
- Supports baud rates up to 115200 bps
- Utilizes USB CDC (Communication Device Class)

3. System Testing

3.1 Testing Methodology

I noticed Node-Red dashboard components shoes the real time values on Temperature, Humidity, and motion. I could check that how about the changes of that values when I made some kind of change near to each sensor.

I noticed the humidity reading was changed when I covered the sensor by hand or make a air flow by mouth to the sensor. I also noticed about brightness sensor, that its values was been different when I light up phone flasher near to the sensor. And also I noticed when move the motion sensor that readings was changed to True from False.

Every time I have tested the values of inside the laboratory. So sometimes I noted that values are wrong due to overheating the sensors when long time using.

3.2 Data Visualization

I made some graphical outputs with Node-RED dashboard. It has more features to express live data readings of sensors.

4. Summary

4.2 Project Analysis

- Overall project success assessment

I explored about this exercise and successfully make a programme using Arduino and Raspberry Pi, I recommend that anyone can make a low cost and high efficiency measurement system.

- Challenges encountered.

Node-RED functions and Raspberry Pi operations are totally new for me. That was some kind of challenge to complete this exercise. I explore more things via internet and teacher guided me to complete more challenging parts.

- Learning outcomes

I identified about theoretical concepts and as well as practical usage of Arduino and Raspberry Pi technologies.

4.3 Improvements and Future Work

I realized that with this Raspberry Pi, I can link some more other databases with our nodes. It helps to make more detailed dashboard and it helps analyse data and express clear idea about final outcomes.