**Documentation**

*Group i*

*28.06.2024*

# Team members

*JIBAN-UL AZAM CHOWDHURY SHAFIN*

# Introduction

**Internet of Things (IoT) and Wireless Sensors Network in This Project:**

**Internet of Things (IoT):**  
IoT means connecting everyday devices to the internet so they can share and receive data.

In this project, IoT allows me to connect the Arduino devices, which collect both temperature and weight data, to the internet. This way, I can monitor these values from anywhere using my phone or computer.

**Wireless Sensors Network (WSN):**WSN refers to a group of sensors that communicate wirelessly to collect and share data.

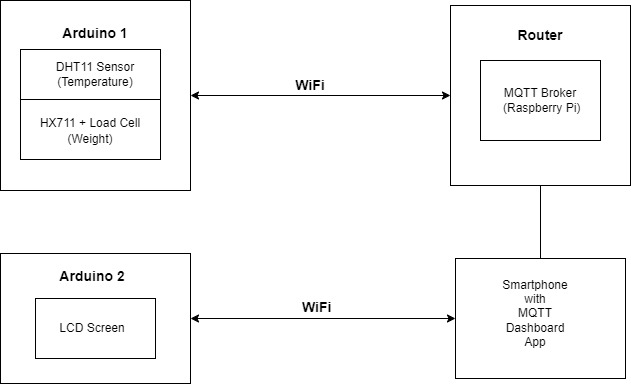
In this project, the temperature sensor (DHT11) and the weight sensor (HX711) on the Arduino send data wirelessly to my MQTT broker, which acts like a central hub for data collection.

**Target Application of This Project:**

The main goal of this project is to monitor temperature and weight remotely:  
  
I am using Arduino devices to read temperature data from a DHT11 sensor and weight data from an HX711 sensor.  
This data is sent wirelessly to an MQTT broker, which acts like a central hub.  
I can then view the temperature on an LCD screen connected to another Arduino.  
I can also receive notifications on my phone using an MQTT app, like IoT MQTT Dashboard, to stay updated on the temperature and weight readings.

.

# Concept description



**Main Application for the Prototype:**

The main application for this prototype is remote environmental monitoring.

Actually, it is used for:  
  
Monitoring temperature and weight in a given environment.  
Providing real-time data to a central hub (Raspberry Pi acting as an MQTT broker).  
Allowing remote viewing and alerts via a smartphone app.

**Devices, Sensors, Actuators, and Apps Used:**

**For placing the weight I created a *Getränkehalter*** by using cad solidworks and printed it successfully. It helped in the project and fully functional.

**Arduino Uno:** Acts as the main controller for sensors and communication.

**Sensors:**  
DHT11 Sensor: Measures temperature.  
HX711 Load Cell: Measures weight.

**Actuators and Display:**  
  
LCD Screen (I2C): Displays the temperature data received from the MQTT broker.

**Communication Modules:**  
WiFi Module (Arduino Uno WiFi Rev2 boards): Facilitates wireless communication.

MQTT Broker:  
  
Raspberry Pi Acts as the MQTT broker to receive and distribute sensor data.

**Software and Libraries:**  
  
Arduino IDE: Used to write and upload code to Arduino.  
PubSubClient Library: Used for MQTT communication.  
Wire Library: Used for I2C communication with the LCD.  
Smartphone App: IoT MQTT Dashboard: An app for receiving and displaying MQTT messages on a smartphone.

# Project/Team management

**Project Methodology Used: Agile Methodology**

**1. Breakdown:**   
  
**Task Breakdown:** The project began with a comprehensive breakdown of requirements into manageable tasks. These tasks included hardware setup, software development, integration, testing, and documentation.  
  
**Sprints:** I structured my work into iterative sprints, typically spanning several days to a week. Each sprint had clear objectives and deliverables, such as setting up sensors, establishing MQTT communication, or developing specific functionalities.  
  
**Daily Stand-ups:** To keep track of progress and address challenges promptly, I conducted informal daily stand-ups (self-check-ins). This helped maintain focus, prioritize tasks, and adjust plans as needed based on daily achievements and obstacles encountered.  
  
**2. Different Tasks/Roles of the Team Members in the Project:**  
**Solo Execution:** As the sole contributor, I assumed all roles and responsibilities throughout the project lifecycle.  
  
**Task Allocation:** Tasks were allocated based on dependencies and critical path analysis. For instance, initial phases focused on hardware setup and sensor calibration before progressing to software development and integration.  
  
**Development and Testing:** I handled the end-to-end development process, from writing Arduino sketches and configuring MQTT protocols to testing sensor accuracy, data transmission reliability, and user interface responsiveness.  
  
**Documentation and Reporting:** Comprehensive documentation was maintained throughout, encompassing project goals, methodologies, wiring diagrams, code snippets, test results, and troubleshooting procedures. This ensured clarity, replicability, and future scalability of the project.  
  
**3. Description of Tasks Performed:**  
  
**Hardware Setup and Integration:** Initially, I configured Arduino boards with necessary sensors (DHT11, HX711 Load Cell), ensuring correct wiring and functionality. Integration involved establishing communication protocols between sensors, Arduino, and peripheral devices like LCD screens.  
  
**Software Development:** Using Arduino IDE and libraries like PubSubClient and Wire, LCD\_I2C, WIFI NINA, I developed firmware to handle sensor data acquisition, MQTT messaging, and LCD display updates. Code optimization and debugging were critical to ensure efficient performance and reliable data transmission.  
  
**Integration and Testing:** System integration involved validating hardware-software interactions and overall system functionality. Testing methodologies, including unit testing for individual components and system testing for integrated functionalities, ensured reliability.  
  
**Continuous Improvement:** Throughout the project, I maintained a focus on continuous improvement. Iterative feedback loops allowed for enhancements in sensor calibration algorithms, MQTT message handling efficiency, and user interface responsiveness based on real-world testing and user feedback*.*

# Technologies

**The technological approaches used in implementing the IoT project:**  
  
**Sensor Technologies:**  
**Temperature Sensor (DHT11):** Utilized to measure ambient temperature with reasonable accuracy and simplicity. It communicates with the Arduino using a single-wire digital interface.  
  
**Weight Sensor (HX711 Load Cell):** Employed to measure weight accurately. It operates by converting mechanical force into an electrical signal, which is then amplified and digitized by the HX711 module for precise weight readings.  
  
**Communication Protocols:**  
**MQTT (Message Queuing Telemetry Transport):** Chosen as the communication protocol between the Arduino and the MQTT broker (hosted on a Raspberry Pi). MQTT is lightweight and efficient, ideal for IoT applications where low bandwidth and power consumption are crucial factors.

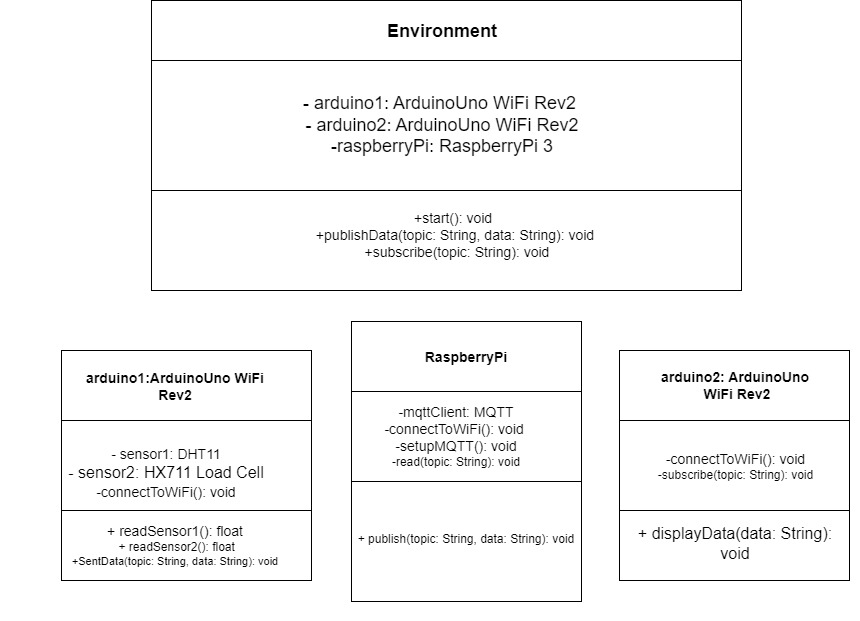
**Programming Languages:**  
**Arduino (C/C++ with Arduino IDE):**  
  
C/C++: The firmware for the Arduino Uno is developed using C/C++. This language is well-suited for microcontroller programming due to its efficiency, direct hardware access, and extensive library support provided by the Arduino ecosystem.

Arduino IDE: Used as the primary integrated development environment for writing, compiling, and uploading firmware to the Arduino Uno. It simplifies the development process with its user-friendly interface and libraries tailored for sensor interfacing and communication protocols.

**Raspberry Pi (Python):**  
Python: The Raspberry Pi hosts the MQTT broker (Mosquitto) and runs Python scripts to interact with MQTT messages and potentially other tasks like data processing or forwarding.

**Implementation Strategy:**  
  
**Arduino Development:** Involves writing C/C++ code within the Arduino IDE to initialize sensors (DHT11 and HX711 Load Cell), read sensor data, format MQTT messages, and establish MQTT client connectivity. This code ensures the Arduino can publish sensor data to predefined MQTT topics.  
  
**Raspberry Pi Development:** Python scripts are written and executed on the Raspberry Pi. These scripts subscribe to MQTT topics, receive data published by the Arduino, and perform necessary actions based on the received data. The use of Python facilitates rapid development, effective handling of MQTT messaging.

# Implementation

**Class Diagram:**

**Explanation of Revised Classes/Modules:**

**Environment:**Represents the overall system.

**Attributes:**

arduino1: Instance of ArduinoUnoWiFiRev2 class.  
arduino2: Instance of ArduinoUnoWiFiRev2 class.  
raspberryPi: Instance of RaspberryPi class.  
Methods:  
start(): Initializes and starts the environment.  
publishData(topic, data): Publishes data to a specified MQTT topic.  
subscribe(topic): Subscribes to a specified MQTT topic to receive data.

1st ArduinoUnoWiFiRev2:  
Represents 1st Arduino Uno WiFi Rev2.

Attributes:  
sensor1: Instance of DHT11 representing the temperature and humidity sensor.  
sensor2: Instance of HX711 Load Cell representing the weight sensor.

connectToWiFi(): Connects Arduino Uno WiFi Rev2 to the WiFi network.

Methods:  
readSensor1(): Reads temperature and humidity data from DHT11 sensor.  
readSensor2(): Reads weight data from HX711 Load Cell sensor.

SentData(topic, data): send data to a specified MQTT topic.

2nd ArduinoUnoWiFiRev2:  
Represents 2nd Arduino Uno WiFi Rev2.

Attributes:

connectToWiFi(): Connects Arduino Uno WiFi Rev2 to the WiFi network.  
subscribe(topic): Subscribes to a specified MQTT topic to receive data.

Methods:  
lcd: presenting the data to the LCD connected to 2nd Arduino Uno WiFi Rev2.

RaspberryPi: Represents a Raspberry Pi microcomputer.

Attributes:  
mqttClient: Instance managing MQTT communication.

setupMQTT(): Sets up MQTT client for communication.

Read(topic): Read data sent from 1st Arduino .

Methods:  
publishData(topic, data): Publishes data to a specified MQTT topic.  
  
**Revised Use Case:**

**Use Case: Monitoring and Displaying Sensor Data**  
  
Actor: User  
  
Description: The system monitors temperature and weight data using Arduino Uno and Arduino Uno WiFi Rev2 devices equipped with DHT11 and HX711 Load Cell sensors, respectively. The Raspberry Pi subscribes to MQTT topics where the Arduino devices publish their sensor readings. Upon receiving data, the Raspberry Pi processes and it is displayed on an LCD screen connected to 2nd Arduino uno wifi rev2 .  
Steps:  
  
Arduino Uno collects temperature data (DHT11) and publishes it to MQTT topic sensor/temperature.  
Arduino Uno WiFi Rev2 collects weight data (HX711 Load Cell) and publishes it to MQTT topic sensor/weight.  
Raspberry Pi subscribes to MQTT topics sensor/temperature and sensor/weight.  
Upon receiving temperature data, it is displayed it on the LCD connected to 2nd Arduino Uno WiFi Rev2.  
Upon receiving weight data, it is displayed it on the LCD connected to 2nd Arduino Uno WiFi Rev2.

Upon receiving the data it is also showed in the phone by subscribing the topic in the app Iot Mqtt Panel

User observes real-time sensor data displayed on the respective LCD screens.

**Video record: https://drive.google.com/file/d/1OgrPlmLL7kCTPEYb5Meetd-5-1TM5KI2/view?usp=sharing**

# Use Case

**How to Use the IoT Application:**

**Setup and Placement**

**Place the Product:**  
Place the product or food item on the "Gatrankhalter" (assuming it's a storage or weighing platform connected to Arduino Uno WiFi Rev2).

**Normal Use:**  
Use the product as usual. The temperature of the product will be continuously displayed on the LCD connected to Arduino Uno WiFi Rev2.

Monitoring from Mobile App  
IoT MQTT Dashboard App:Download and install the "IoT MQTT Dashboard" app on your smartphone (available for iOS and Android).

Open the app and configure it to connect to the MQTT broker using the IP address (172.20.10.5) and port (1883).

**Subscribing to Topics:**  
  
**Subscribe to the MQTT topics:**  
sensor/temperature: This topic displays the current temperature of the product.  
sensor/weight: This topic indicates the weight of the product.

**Viewing Real-Time Data:**  
  
Once subscribed, you can view real-time temperature and weight data from your smartphone.  
This allows you or other authorized users to monitor the conditions of the product remotely.

**Notifications for Low Food Weight**  
**Setting Thresholds:**  
Configure a threshold weight level in your Arduino Uno WiFi Rev2 code.  
When the weight of the product drops below this threshold, Arduino Uno WiFi Rev2 will send a message (sensor/weight topic) to the MQTT broker.

**Receiving Notifications:**  
  
Users subscribed to the MQTT topic **sensor/weight** will receive notifications on their mobile devices.  
These notifications alert the user that the food is running low and needs replenishment.