SAVONIA
University of Applied Sciences

THESIS – BACHELOR'S DEGREE PROGRAMME
TECHNOLOGY, COMMUNICATION AND TRANSPORT

Implementing NB-IoT for Voltage and Current Measurement with Arduino and Cloud Integration

AUTHOR/S Radomir Pestov

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology, Internet of Things	
Author(s) Radomir Pestov	
Title of Thesis Implementing NB-IoT for Voltage and Current Measurement with Arduino and Cloud Integration	
Date 29.05.2024	Pages/Number of appendices 48
Client Organisation /Partners Savonia-ammattikorkeakoulu	

Abstract

This project aimed to investigate the integration of Narrowband Internet of Things (NB-IoT) technology with Arduino for precise voltage and current measurements to enhance monitoring systems with cloud connectivity. The study was commissioned to explore the viability of utilizing NB-IoT in IoT applications for improved sensor-based systems.

The methodology encompassed the utilization of Arduino Mega and the DFR0763 NB-IoT module, outlining the circuit design and data transmission to Thingspeak. The study evaluated the accuracy of voltage and current measurements, explored cloud integration possibilities, and compared the data processing capabilities of platforms like Thingspeak and Azure.

The key findings highlighted the successful establishment of a functional system for precise measurements, validated through a comprehensive evaluation. The study's major conclusion emphasizes the practical significance of NB-IoT integration in enhancing monitoring systems, offering insights into improved sensor-based data collection and cloud connectivity.

Keywords

NB-IoT, Arduino, Voltage and Current Measurements, Cloud Integration, Data Transmission

CONTENTS

LIST OF FIGURES

Acknowledgements

I would like to express my gratitude to several individuals and organizations for their support and contributions to this thesis.

Firstly, I am deeply grateful to my supervisors, Markku Kellomäki and Arto Toppinen, for their unwavering support, insightful feedback, and valuable guidance throughout the research process.

Furthermore, I would like to thank my colleagues and friends for their encouragement and constructive discussions that enriched this research.

Lastly, I am profoundly thankful to my family for their unconditional support and understanding during the entire period of my study.

1 INTRODUCTION

The rapid evolution of the Internet of Things (IoT) has ushered in a new era of connectivity, changing how we collect, analyze, and utilize data from the physical world. In this context, the Narrowband Internet of Things (NB-IoT) (Chen 2017, 20557) is becoming a key technology, offering enhanced capabilities for efficient and reliable communication in IoT applications. This thesis explores integrating NB-IoT technology with Arduino for precise voltage and current measurements, specifically focusing on improving monitoring systems through cloud connectivity.

The beginning of this research traces back to an enriching internship opportunity provided to me by my teacher at Savonia. The internship immersed me in a project that delved into the capabilities of the NB-IoT module for voltage measurement and the seamless transmission of this data to the cloud for in-depth analysis. As the project progressed, I became more and more fascinated by its complexities and the countless possibilities it opened.

This thesis is a dedicated effort to delve deeper into the functionality and practical implementation of the NB-IoT module, particularly within the domain of voltage measurement. It seeks to comprehend and showcase the efficacy of this technology in real-world applications, specifically in transmitting voltage data to the cloud for subsequent analysis and comparison.

The motivation behind this work lies in addressing contemporary challenges in remote power measurements, such as those in solar and wind energy systems, and tackling power quality issues in various electrical installations. Remote monitoring of power systems is crucial for optimizing performance, predicting maintenance needs, and ensuring the reliability of energy supplies. For instance, in renewable energy applications, precise voltage and current measurements are essential for monitoring the output of solar panels and wind turbines, facilitating efficient energy management and integration into the grid. Furthermore, understanding power quality issues such as voltage sags, surges, and harmonic distortions can help maintain the health of electrical systems and prevent potential damage to sensitive equipment.

By leveraging NB-IoT technology, this research aims to provide a robust solution for real-time monitoring and analysis of electrical parameters, enhancing the ability to manage and optimize power systems remotely. The integration of cloud connectivity enables continuous data collection and sophisticated analysis, ultimately contributing to more resilient and efficient energy infrastructures.

Objectives of the Thesis:

Functionality Assessment: Investigate the effectiveness of the NB-IoT module in facilitating precise voltage measurements, exploring inherent challenges and limitations.

Cloud Integration Analysis: Examine the procedures for transmitting voltage data to the cloud using the NB-IoT module. Evaluate the performance of data transmission and reception in practical scenarios.

Comparison of Cloud Platforms: Assess the capabilities of the chosen cloud platform (Thingspeak) in handling and analyzing transmitted voltage data. Compare strengths and weaknesses with alternative platforms like Azure.

Practical Applicability: Uncover practical insights and implications from implementing NB-IoT for voltage measurement. Explore how these findings can be extrapolated to real-world IoT and monitoring systems applications.

Research Questions:

How effectively does the NB-IoT module facilitate voltage measurement, and what are the inherent challenges or limitations?

What procedures are involved in transmitting voltage data to the cloud using the NB-IoT module? How does the data transmission and reception perform in practical scenarios?

How does the chosen cloud platform (Thingspeak) fare in handling and analyzing the transmitted voltage data? What are its strengths and weaknesses compared to alternative platforms like Azure?

What practical insights and implications emerge from implementing NB-IoT for voltage measurement? How can these findings be extrapolated to real-world IoT and monitoring systems applications?

During writing this thesis, I utilized ChatGPT (ChatGPT 2024.), a language model (GPT-4o, May 2024) developed by OpenAI (OpenAI 2024.), for assistance with language checking and refinement. ChatGPT helped me ensure that the content of the thesis is clear, coherent, and professionally presented. I used the tool to review my writing for grammatical accuracy, improve sentence structure, and enhance overall readability. This integration of AI tools showcases the practical application of advanced technologies not only in technical research but also in academic writing, emphasizing the potential benefits of AI in improving the quality and clarity of academic documents.

2 LITERARURE REVIEW

2.1 Overview of IoT and its applications in monitoring systems

The Internet of Things (IoT) has emerged as a transformative paradigm, connecting physical devices and enabling them to communicate, share data, and operate collaboratively. In the realm of monitoring systems, IoT offers unparalleled capabilities for real-time data collection, analysis, and decision-making. Monitoring systems, ranging from environmental sensing to industrial processes, benefit from the seamless integration of IoT technologies. (Atzori 2010, 2787.)

IoT in Monitoring Systems: Revolutionizing Data Collection and Analysis IoT technologies have revolutionized traditional monitoring systems by enabling the integration of sensors, actuators, and communication networks. These interconnected systems facilitate the continuous monitoring of physical parameters such as temperature, humidity, pressure, and motion. By collecting and analyzing data in real-time, IoT-enabled monitoring systems provide valuable insights into the performance, status, and behavior of monitored assets. (Da Xu 2014, 2233-2243.)

The applications of IoT in monitoring systems span across diverse domains, showcasing its versatility and adaptability. In healthcare, IoT devices are utilized for remote patient monitoring, medication adherence tracking, and healthcare asset management. These systems enable healthcare providers to deliver personalized care, monitor patient vital signs, and detect anomalies in real-time, thereby improving patient outcomes and reducing hospital readmissions. In agriculture, IoT-based monitoring systems are revolutionizing precision farming practices. Sensors deployed in fields collect data on soil moisture levels, temperature, and nutrient levels, enabling farmers to optimize irrigation schedules, fertilization practices, and crop management strategies. Farmers can increase crop yields, conserve water resources, and mitigate environmental impact by leveraging IoT technologies. Smart cities leverage IoT to monitor and manage critical infrastructure and public services. IoT sensors embedded in urban environments monitor air quality, traffic congestion, waste management, and energy consumption. This data is used to optimize city operations, enhance public safety, and improve the overall quality of life for residents. Smart city initiatives also enable efficient transportation systems, intelligent parking solutions, and sustainable urban development practices. In industrial automation, IoT-enabled monitoring systems play a pivotal role in enhancing operational efficiency and safety. Industrial IoT (IIoT) platforms integrate sensors, actuators, and control systems to monitor equipment health, predict maintenance needs, and optimize production processes. By leveraging real-time data analytics and predictive maintenance algorithms, industrial organizations can minimize downtime, reduce maintenance costs, and ensure worker safety. (Al-Fuqaha 2015, 2347-2376.)

The ability to remotely monitor and control devices has led to significant benefits and impact across various sectors. By harnessing the power of IoT technologies, organizations can achieve: - Improved Efficiency: Real-time monitoring enables proactive decision-making, optimizing resource utilization and reducing wastage. - Cost Reduction: Predictive maintenance and asset monitoring help minimize downtime and maintenance costs, leading to overall cost savings. - Enhanced Safety: Continuous monitoring of environmental conditions and equipment performance enhances workplace safety and

reduces the risk of accidents. - Data-Driven Insights: IoT-generated data provides valuable insights for process optimization, product innovation, and strategic decision-making. (Manyika 2015.)

The integration of IoT technologies into monitoring systems has unlocked new possibilities for data collection, analysis, and decision-making across various domains. From healthcare and agriculture to smart cities and industrial automation, IoT-enabled monitoring systems are revolutionizing how we monitor and manage the world around us. IoT solutions have significantly enhanced the efficiency and reliability of energy generation, distribution, and consumption in the energy sector. For instance, smart grids utilize IoT devices to monitor and manage the flow of electricity, ensuring optimal performance and reducing energy losses. In renewable energy, IoT-enabled sensors and monitoring systems are used to track the performance of solar panels and wind turbines, optimizing energy output and maintenance schedules. Additionally, IoT-based energy management systems in buildings and industrial facilities enable real-time monitoring and control of energy usage, leading to substantial cost savings and reduced environmental impact. As technology continues to evolve, the potential for IoT to drive innovation and create intelligent, responsive monitoring ecosystems remains limitless. (Gubbi 2013, 1645; Mehmood 2017, 16; Alahakoon 2015, 425-436.)

2.2 Explanation of NB-IoT technology and its advantages in IoT applications

The Narrowband Internet of Things (NB-IoT) technology represents a significant advancement in IoT communications, tailored specifically to meet the demands of low-power, wide-area networks (LPWANs). It offers a specialized communication standard designed to provide reliable connectivity for IoT devices while optimizing energy consumption and ensuring cost-effectiveness. (Raza 2017, 855.)

NB-IoT operates on existing cellular networks, utilizing licensed spectrum bands to establish communication links between IoT devices and network infrastructure. Unlike traditional cellular networks, NB-IoT is optimized for scenarios where devices require intermittent communication over long distances with minimal power consumption. This optimization makes NB-IoT particularly suitable for applications involving remote monitoring, asset tracking, and environmental sensing. (Yin 2022, 838-842.)

The architecture of NB-IoT (Figure 1) comprises several key components:

1. NB-IoT Devices: These are the end devices or sensors that collect data and communicate it to the network. They are designed to be energy-efficient and can operate on battery power for extended periods.
2. NB-IoT Base Stations: These base stations are part of the existing LTE infrastructure and are modified to support NB-IoT. They handle the communication between NB-IoT devices and the core network.
3. NB-IoT Core Network: The core network, which includes components such as the Mobility Management Entity (MME) and the Serving Gateway (SGW), manages the data routing, device authentication, and service delivery.
4. NB-IoT Cloud Platform: These servers process the data received from NB-IoT devices and provide services such as monitoring, analytics, and control.

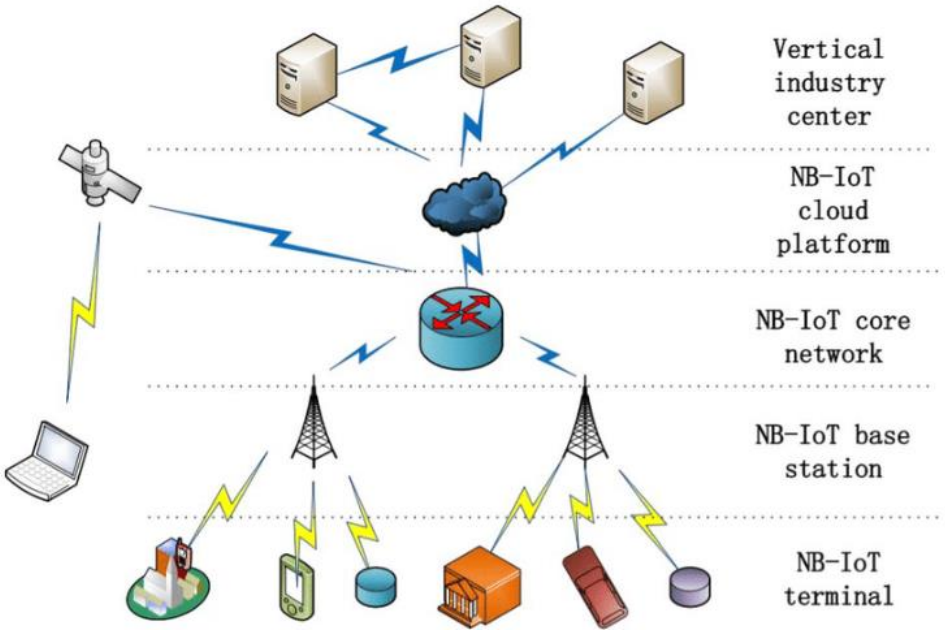

Figure 1. NB-IoT Architecture

Comparison with Other Wireless Technologies

To better understand the advantages and limitations of NB-IoT, it's essential to compare it with other alternative wireless technologies commonly used in IoT applications:

LoRa (Long Range) (Devalal 2018, 284-290.)

- Range: LoRa offers similar range capabilities to NB-IoT, often reaching several kilometers in urban environments and up to 15 kilometers in rural areas.
- Frequency Band: Operates in unlicensed spectrum bands, which can lead to interference issues.
- Data Rate: It generally has lower data rates than NB-IoT, making it suitable for applications requiring small amounts of data.
- Power Consumption: Very low power consumption, enabling long battery life.
- Deployment: Requires deployment of proprietary gateways and network infrastructure.

LTE-M (LTE Cat-M1) (Lauridsen 2016, 1-5.)

- Range: Like NB-IoT, leveraging existing LTE networks.
- Frequency Band: Uses licensed spectrum, providing reliable and interference-free communication.
- Data Rate: Higher data rates than NB-IoT, suitable for applications requiring more bandwidth.
- Power Consumption: Slightly higher power consumption than NB-IoT but still suitable for battery-operated devices.
- Deployment: Uses existing LTE infrastructure, simplifying deployment for operators.

Wi-Fi (Ma 2019, 1-36)

- Range: Limited range compared to NB-IoT, typically up to 100 meters indoors.
- Frequency Band: Operates in unlicensed spectrum bands, susceptible to interference.
- Data Rate: High data rates, suitable for applications requiring substantial bandwidth.
- Power Consumption: High power consumption, not ideal for battery-operated devices.
- Deployment: Widely available infrastructure but limited to areas with Wi-Fi coverage.

While each wireless technology has its strengths and weaknesses, NB-IoT stands out for its optimal balance of range, power consumption, and data rate, making it particularly suitable for applications involving remote monitoring, asset tracking, and environmental sensing. The choice of technology ultimately depends on the application's specific requirements, including data rate, power consumption, range, and deployment considerations. In conclusion, NB-IoT selection for this project is driven by its ability to provide reliable, long-range communication with minimal power consumption, leveraging existing cellular infrastructure to ensure broad coverage and seamless integration into IoT ecosystems.

Advantages of NB-IoT in IoT Applications

1. Extended Coverage:
   One of the primary advantages of NB-IoT is its ability to penetrate deep into buildings and underground structures, providing connectivity in challenging environments where traditional cellular signals may struggle to reach. This extended coverage ensures that IoT devices remain connected even in remote or hard-to-reach locations, enhancing the reliability of data transmission. (Adhikary 2016, 1-5.)
2. Low Power Consumption:
   NB-IoT devices are designed to operate on minimal power, making them ideal for battery-powered or energy-constrained applications. By employing power-saving mechanisms such as extended discontinuous reception (eDRX) and power-saving mode (PSM), NB-IoT devices can achieve long battery life while maintaining continuous connectivity with the network. (Raza 2017, 1-36.)
3. Cost-Effectiveness:
   The deployment of NB-IoT networks leverages existing cellular infrastructure, minimizing the need for additional infrastructure investments. This cost-effective approach enables network operators to expand coverage and support many IoT devices without incurring significant capital expenditures. (Sinha 2017, 14-21.)
4. Suitability for Massive Deployments:
   NB-IoT technology is well-suited for scenarios requiring deploying many IoT devices within a geographical area. Its efficient use of spectrum resources and robust communication protocols enable seamless connectivity for large-scale IoT deployments, such as smart city initiatives, industrial monitoring systems, and agricultural sensor networks. (Palattella 2016, 510-527.)

NB-IoT technology offers compelling advantages for IoT applications as shown in the list. Its specialized communication standard fills a crucial niche within the IoT ecosystem, enabling reliable connectivity for various devices and use cases. As research and development in NB-IoT continue to progress, the technology is poised to play a significant role in shaping the future of IoT connectivity and infrastructure. (Malik 2022, 172.)

2.3 Review of similar projects involving Arduino, sensor measurements, and cloud integration

Several projects documented in the literature have delved into integrating Arduino microcontrollers with various sensors, emphasizing data acquisition and cloud integration. These endeavors typically showcase Arduino's adaptability in interfacing with various sensors, including those for measuring voltage, current, temperature, humidity, and more. The primary objective across these projects is to harness Arduino's capabilities for precise sensor readings and then seamlessly transmit this data to cloud platforms for remote monitoring, storage, and analysis. (Debele 2020, 428-432.)

In these projects, researchers commonly select sensors based on the specific parameters they aim to measure, such as voltage and current for electrical systems or environmental parameters for weather monitoring. They then develop Arduino code to interface with these sensors, ensuring accurate data acquisition and conversion. Communication protocols, such as MQTT or HTTP, are established to facilitate the transmission of sensor data to cloud platforms like Thingspeak, AWS, or Azure. (Pulver 2019)

Researchers assess the effectiveness of these integrated systems through real-world experimentation, evaluating factors such as reliability, scalability, and ease of implementation. Reliability pertains to the consistency and accuracy of sensor readings, while scalability refers to the system's ability to accommodate increasing numbers of sensors or users. Ease of implementation considers the simplicity of setting up and maintaining the integrated hardware and software components.

These projects contribute significantly to advancing IoT applications by demonstrating practical implementations of sensor data acquisition and cloud integration. They serve as valuable references for developers and researchers seeking to build similar systems for diverse monitoring and automation purposes. Moreover, they highlight the potential of Arduino-based solutions in enabling cost-effective and accessible IoT deployments across various domains. (Zafar 2018, 3238-3242.)

2.4 Examination of existing methodologies for voltage and current measurement

The accurate voltage and current measurement is crucial in IoT applications, especially in systems where electrical parameters play a vital role. The existing literature extensively covers various methodologies employed to achieve precise voltage and current measurements, encompassing both traditional analog techniques and modern digital approaches.

Analog-to-digital converters (ADCs) (Walden 1999, 539-550.) are commonly used in IoT systems to convert continuous analog signals, such as voltage and current, into digital data that can be processed by microcontrollers or computers. These ADCs sample the analog signal at regular intervals and quantize it into discrete digital values, allowing for accurate measurement and analysis.

Additionally, sensor calibration techniques are employed to calibrate sensors used for voltage and current measurements, ensuring their accuracy and reliability over time. (Ripka 2010, 1108-1116.)

Modern digital approaches utilize advanced signal processing algorithms and noise reduction strategies to enhance the precision of voltage and current measurements. Digital filters, such as finite impulse response (FIR) filters and infinite impulse response (IIR) filters, are applied to remove unwanted noise and artifacts from the measured signals, resulting in cleaner and more accurate data. (Litwin 2000, 28-31.)

The literature highlights the challenges associated with voltage and current measurements in IoT applications, including noise interference, signal attenuation, and sensor inaccuracies. Researchers propose various mitigation strategies such as shielding sensitive circuitry, implementing high-quality sensors, and employing signal conditioning techniques to amplify weak signals and minimize noise. (Sigrist 2017, 1159-1164.)

3 METHODOLOGY

3.1 Description of hardware components used

The fundamental hardware components employed in this project play a crucial role in enabling precise voltage and current measurements using Narrowband Internet of Things (NB-IoT) technology. The Figure 2 represents the primary microcontroller chosen for the task - Arduino Mega (Badamasi 2014, 1-4.), a versatile and widely used platform known for its compatibility with various sensors and modules. To facilitate NB-IoT communication, the DFRobot_SIM7000 NB-IoT (DFR-0763) (Dfrobot 2018) module was selected, which is displayed in Figure 3, offering reliable connectivity for transmitting data to the cloud. In addition to the microcontroller and NB-IoT module, the project involves the integration of sensors, specifically a voltmeter and an ammeter. These sensors are essential for measuring voltage and current values from an alternating current (AC) network. The voltmeter is connected to Arduino's analog pin A0, while the ammeter is connected to analog pin A1. This hardware configuration ensures accurate and real-time acquisition of electrical parameters. The chosen hardware components are carefully selected to create a synergistic system capable of capturing precise measurements and transmitting the data efficiently. The combination of Arduino, DFRobot_SIM7000, and the sensors forms the foundation for the successful implementation of NB-IoT technology in voltage and current monitoring. (Kusriyanto 2016, 127-131.)

Figure 2. Arduino Mega

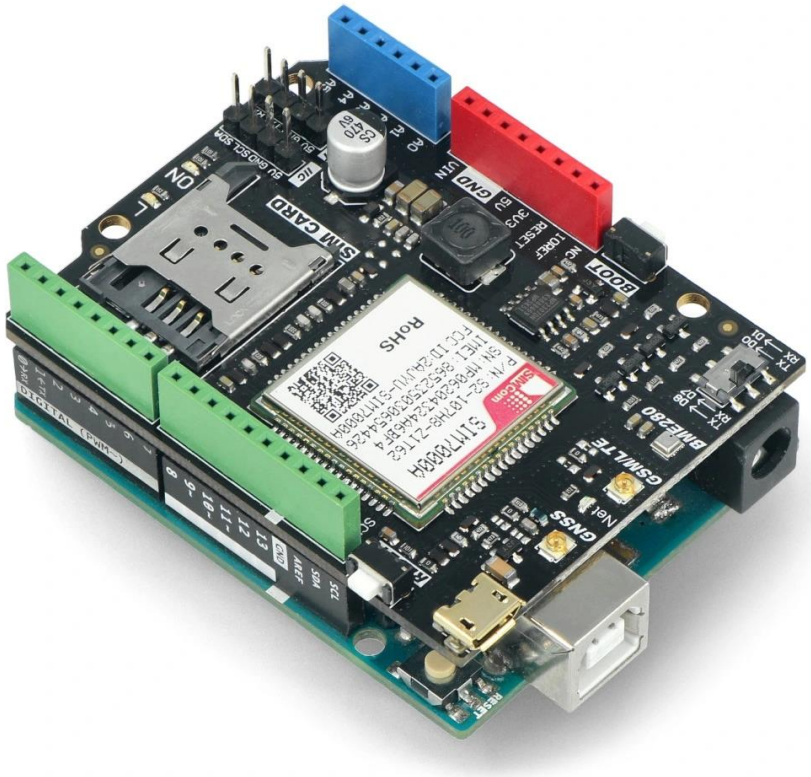

Figure 3. DFR0763 connected to Arduino

3.2 Explanation of the circuit design and connections between components

The circuit design is a critical aspect of the project, dictating how the hardware components interact and ensuring the seamless operation of the NB-IoT-based voltage and current measurement system. The key connections and configurations are designed to optimize performance and accuracy. The serial communication interface between Arduino and the DFRobot_SIM7000 NB-IoT module is established using the SoftwareSerial library (Arduino). This allows communication on pins 7 (RX) and 10 (TX) of Arduino. Additionally, the millis() function is employed to create periodic timer interrupts, set at intervals of 2 milliseconds, triggering the execution of specific functions. The sensors, including the voltmeter and ammeter, are connected to Arduino's analog pins A0 and A1, respectively. This ensures that the analog readings from these sensors are captured efficiently for subsequent processing and transmission. Overall, the circuit design prioritizes clarity, efficiency, and compatibility, laying the groundwork for successfully integrating NB-IoT technology into the voltage and current measurement system (Figure 5).

AC Network Box overview

The AC network box is a crucial component of the voltage and current measurement system. It houses the necessary connections for safely interfacing with the AC power lines. This box ensures that measurements are taken accurately and securely, adhering to safety standards.

Components Inside the AC Network Box:

1. Phase and Neutral Wires:
    - The AC network box has terminals for connecting the phase and neutral wires from the AC main supply.
    - The phase wire is connected to the voltmeter input, and the neutral wire is used both for the voltmeter and the clamp meter (ammeter).
2. Voltmeter:
    - Connected to measure the voltage between the phase and neutral wires, providing the effective line voltage.
    - The voltmeter output is routed through a BNC connector to the level shifter.
3. Ammeter (Clamp Meter):
    - Clamps onto the neutral wire to measure the current flowing through the system without breaking the circuit.
    - The clamp meter output is connected to another level shifter.
4. Level Shifters:
    - Located inside the AC network box, these components scale the voltmeter and ammeter outputs to levels compatible with the Arduino's analog inputs.

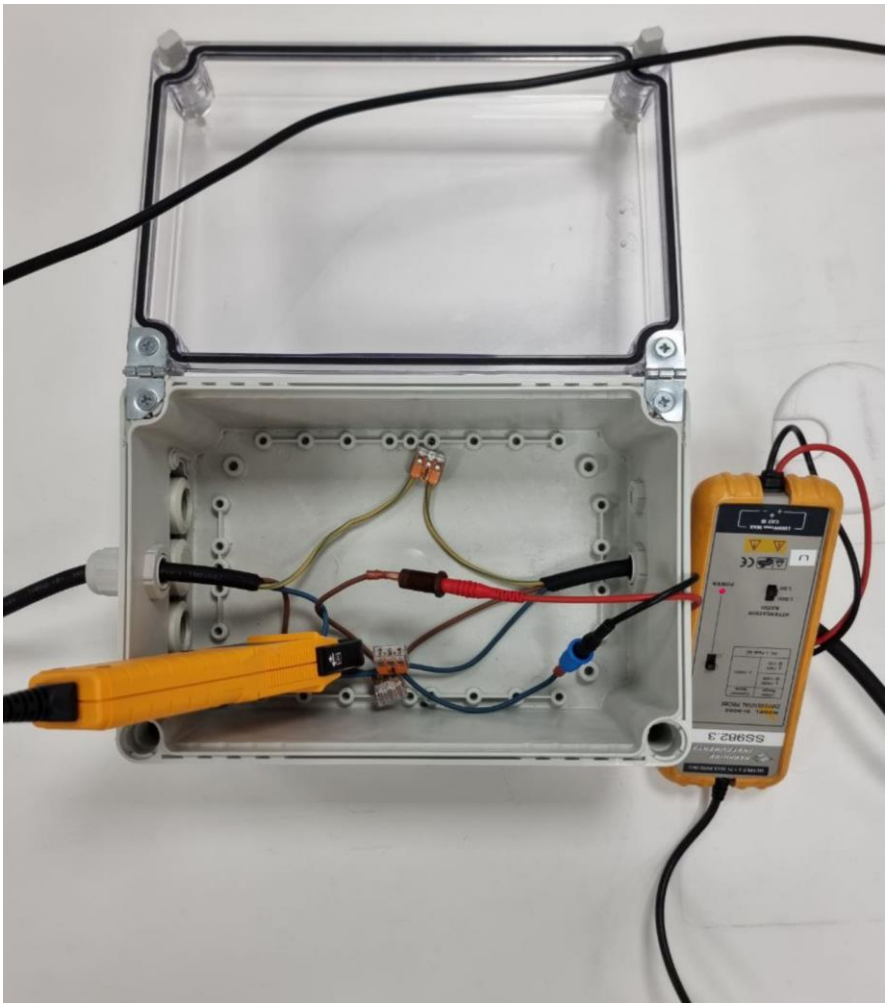

Figure 4. AC Network

Detailed Circuit Overview

1. RTC (DS3231) Setup:
    - The DS3231 RTC is connected to the Arduino Mega via the I2C connection.
    - **Connections:**
        - SDA (RTC) → A4 (Arduino Mega)
        - SCL (RTC) → A5 (Arduino Mega)
    - This setup allows the system to maintain accurate timekeeping for data logging purposes.
2. Voltmeter and Level Shifter Connection:
    - The phase and neutral wires from the AC network are connected to the voltmeter.
    - The voltmeter output is connected to the level shifter using a BNC connector.
    - The level shifter output is connected to the analog pin A0 of the Arduino Mega.
    - **Connections:**
        - Voltmeter → Level Shifter (via BNC)
        - Level Shifter Output → A0 (Arduino Mega)
3. Ammeter (Clamp Meter) and Level Shifter Connection:
    - The clamp meter is used to measure current through the neutral wire.
    - The output from the clamp meter is connected to a level shifter.
    - The level shifter output is connected to the analog pin A1 of the Arduino Mega.
    - **Connections:**
        - Clamp Meter → Level Shifter
        - Level Shifter Output → A1 (Arduino Mega)
4. Serial Communication with NB-IoT Module:
    - The DFRobot_SIM7000 NB-IoT module communicates with the Arduino Mega using the SoftwareSerial library.
    - **Connections:**
        - RX (NB-IoT) → Pin 7 (Arduino Mega)
        - TX (NB-IoT) → Pin 10 (Arduino Mega)
5. Timer Interrupts:
    - The millis() function creates periodic timer interrupts at intervals of 2 milliseconds.
    - These interrupts trigger specific functions for data acquisition and processing.

The integration of the AC network box ensures the safe and accurate measurement of voltage and current, essential for the project's success. This explanation highlights the thoughtful design and engineering behind the NB-IoT-based monitoring system by detailing the connections and providing a visual representation. The combination of Arduino Mega, the DS3231 RTC for timekeeping, and the DFRobot_SIM7000 NB-IoT module for communication ensures a robust and reliable voltage and current measurement solution in IoT applications.

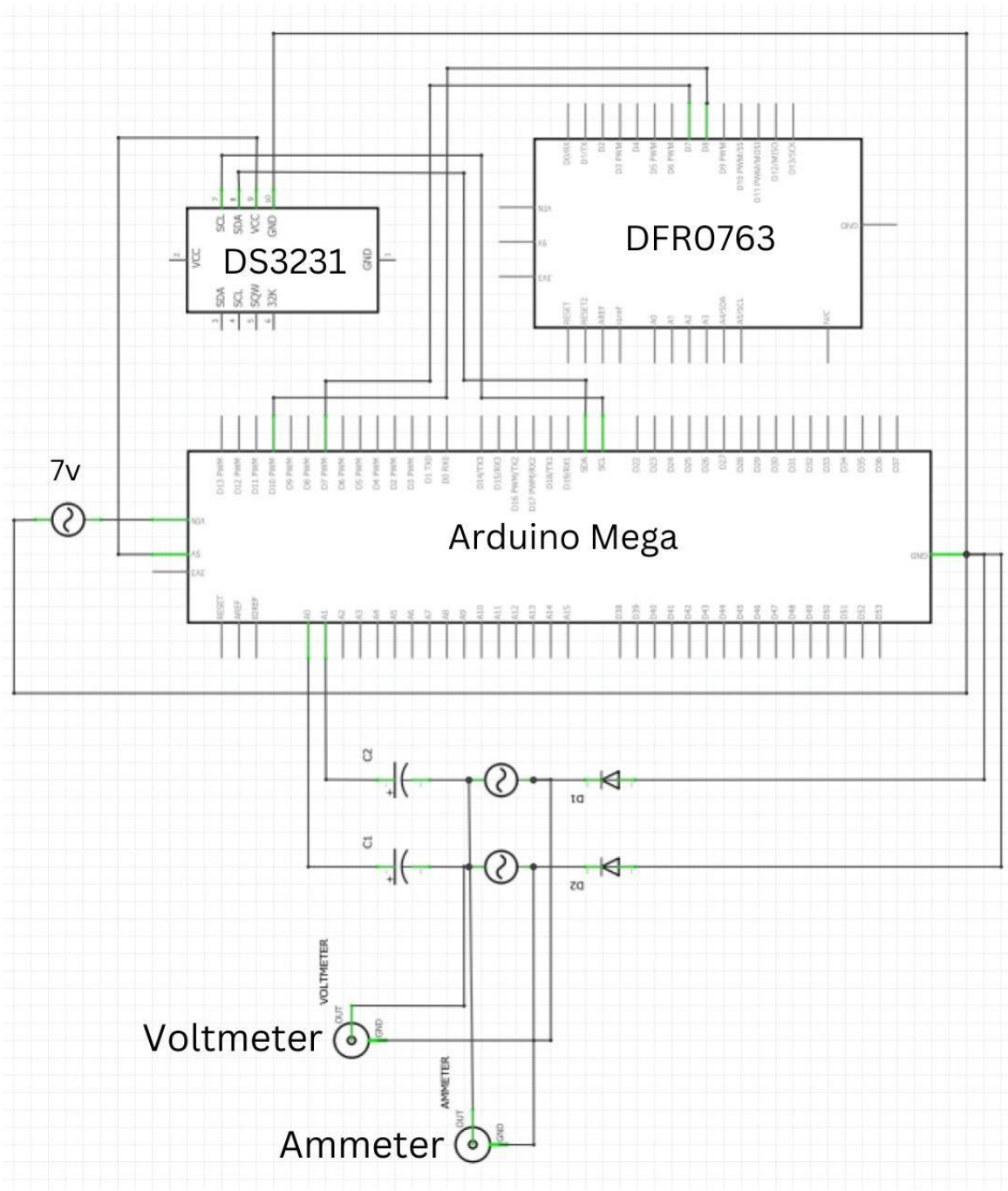

Figure 5. Circuit diagram of connected components

3.3 Integration of RTC DS3231 For Timekeeping

An RTC DS3231 module was incorporated into the project setup in response to the need for precise timekeeping between data samples. The RTC DS3231 offers highly accurate timekeeping capabilities, ensuring that each data point is timestamped precisely.

The RTC DS3231 module interfaces with the Arduino Mega to provide accurate timekeeping functionality. It utilizes a temperature-compensated crystal oscillator (TCXO) to maintain timekeeping accuracy within a few seconds per year, making it an ideal choice for applications requiring reliable timestamping.

The RTC DS3231 module communicates with the Arduino Mega via the I2C protocol, allowing the microcontroller to retrieve current time information and update it as necessary. The module keeps track of time even when the Arduino is powered off, ensuring continuous operation and accurate timestamping of data samples.

The RTC DS3231 module (Figure 6) is connected to the Arduino Mega using the I2C interface. Specific pins on the Arduino (such as SDA and SCL) are utilized for communication with the RTC module, enabling seamless integration into the existing project setup (Analog).

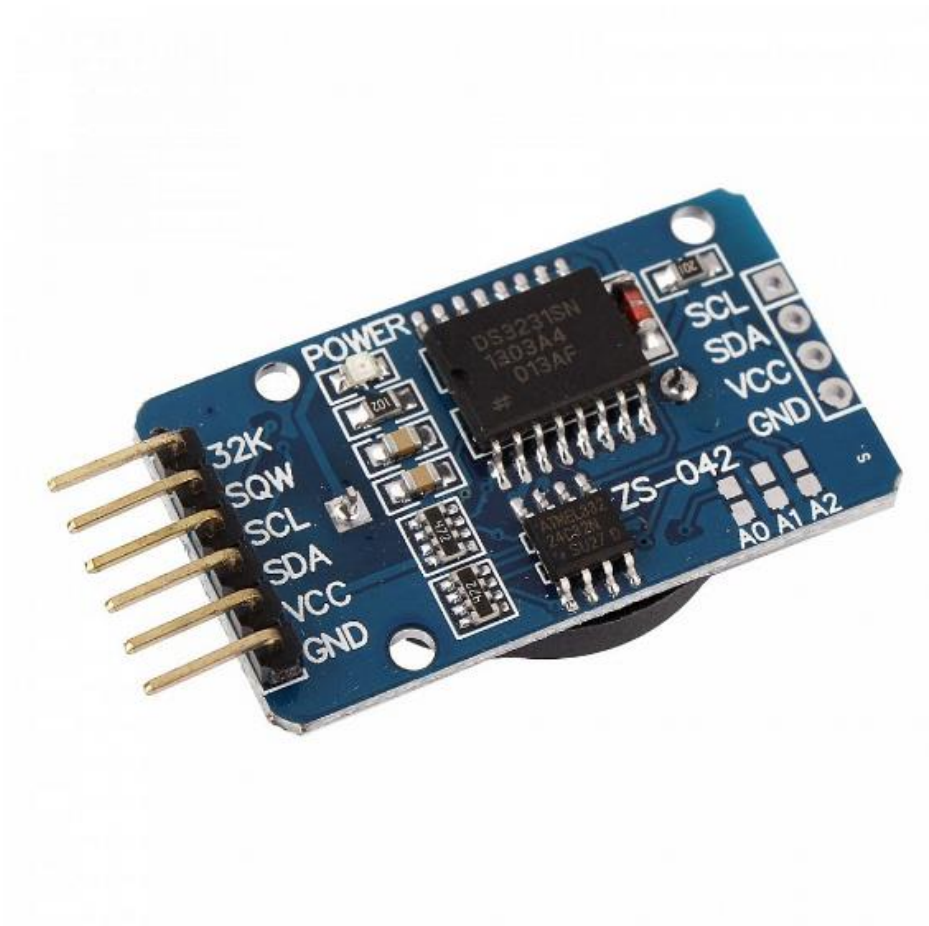

Figure 6. Real Time Clock Module DS3231

3.4 Detailed steps for setting up the NB-IoT module and connecting it to the cloud

The setup of the NB-IoT module and the subsequent connection to the cloud platform involves a series of detailed steps to ensure a robust and reliable implementation. These steps are crucial in establishing a functional system capable of seamlessly transmitting voltage and current data.

SIM7000 Initialization: The initialization process for the DFRobot_SIM7000 NB-IoT module is crucial for establishing reliable communication with the network. This process involves configuring various parameters, including the Access Point Name (APN), which is vital in connecting the module to the Internet via the cellular network.

The Access Point Name (APN) is a configuration setting on the mobile device that identifies an external network that the device can access. It acts as a gateway between the cellular network and

the internet, specifying the network path for all cellular data connectivity. Configuring the correct APN is essential for enabling the NB-IoT module to communicate effectively with the cloud or any other internet services.

PARAMETERS FOR SIM7000 INITIALIZATION

To initialize the SIM7000 module, the following parameters must be configured:

1. APN (Access Point Name):
   - Description: Identifies the network that the module will connect to.
   - Example: internet for a generic APN, but it could vary based on the cellular service provider.
2. Network Selection:
   - Description: Specifies the type of network (NB-IoT, LTE-M, etc.) the module should connect to.
   - Example: NB-IoT for Narrowband IoT.
3. Authentication Type:
   - Description: Determines if authentication is required and which type (PAP, CHAP, etc.).
   - Example: PAP (Password Authentication Protocol).
4. Username and Password:
   - Description: Credentials required by some networks for authentication.
   - Example: Provided by the cellular service provider if needed.
5. Baud Rate:
   - Description: Communication speed between the module and the Arduino.
   - Example: 9600 bps (bits per second).
6. MNO Profile:
   - Description: Mobile Network Operator profile, ensuring compatibility with the network.
   - Example: Specific profile number for the operator.
7. Band Selection:
   - Description: Specifies the frequency bands to be used for communication.
   - Example: B8 (900 MHz) for NB-IoT.

Parameter	AT command	Description	Value
APN	`AT+CGDCONT`	Access Point Name for network connection	`iot.dna.fi`
Network Type	`AT+CNMP`	Specifies the network type (NB-IoT)	`38` (for NB-IoT)
Authentification	`AT+CGAUTH`	Type of authentication (PAP/CHAP)	` 1,1,"username","password"`

Baud Rate	`AT+IPR`	Communication speed between module and Arduino	`9600`
MNO Profile	`AT+UMNOPROF`	Mobile Network Operator profile	`100`
Frequency Band	`AT+NBAND`	Specifies the frequency band for communication	`8` (900MHz)

Timer Setup: Utilizing the millis() function, the project configures millis() to generate periodic interrupts at 2 millisecond intervals. This timer serves as a crucial component in triggering the acquisition of voltage and current readings at regular intervals.

Cloud Platform Integration: The integration with the cloud platform, specifically ThingSpeak in this case, is a key aspect of the project. The sendDataToThingspeak function orchestrates the opening of a network connection to ThingSpeak and the subsequent transmission of HTTP GET requests containing voltage, current, and timestamp data. This ensures that the collected measurements are efficiently communicated to the cloud for further analysis. The detailed steps in setting up the NB-IoT module and connecting to the cloud underscore the importance of a systematic and well-defined approach to achieve seamless communication and data transmission.
ThingSpeak is an open-source Internet of Things (IoT) analytics platform that allows users to collect, store, analyze, visualize, and act on data from sensor networks. It provides an easy-to-use interface for setting up channels to receive data, creating visualizations, and performing real-time analysis. ThingSpeak supports various communication protocols, including HTTP, MQTT, and RESTful APIs, making it versatile for different IoT applications. (Kelechi 2022, 151-169.)

Setting Up ThingSpeak

1. Create an Account:
    - Sign up for a free account on ThingSpeak.
    - Once logged in, create a new channel to store your data.

2. Create a New Channel:
    - Navigate to the Channels tab and click on "New Channel".
    - Configure the channel by adding a name and description.
    - Add fields for the data you will be sending (e.g., Voltage, Current, Timestamp).
    - Save the channel and note the Channel ID and API Keys (Write API Key for sending data and Read API Key for reading data).

3. API Keys:
    - Write API Key: Used to send data to ThingSpeak.

- Read API Key: Used to read data from ThingSpeak.

3.5 Code explanation: Arduino code for data acquisition, processing, and transmission to the cloud

The Arduino code forms the basis of the project, orchestrating the entire process of data acquisition, processing, and transmission to the cloud. The code is designed with clarity, efficiency, and functionality, ensuring that the Arduino Mega operates as a reliable hub for acquiring and transmitting voltage and current data.

takeReading(): This function is responsible for capturing voltage and current values from the Arduino's analog pins. The analog readings are then used to construct a Reading structure, incorporating the voltage, current, and a timestamp obtained through the RTC (Figure 6).

```
Reading takeReading() {
  DateTime now = rtc.now();//set up a time for RTC module
  unsigned long milliseconds = millis();
  float voltage = analogRead(VOLTAGE_PIN) * (5.0 / 1023.0);
  float current = analogRead(CURRENT_PIN) * (5.0 / 1023.0);

  return {now,milliseconds,voltage, current};
}
```

Figure 7. takereading code snippet

printReading(Reading reading): This function facilitates the printing of the captured voltage, current, and timestamp data to the Serial Monitor. This real-time feedback is valuable for monitoring the system's operation and verifying the accuracy of measurements (Figure 7).

```
void printReading(Reading reading) {
  Serial.print("Voltage: ");
  Serial.print(reading.voltage, 2); // Print voltage with 2 decimal places
  Serial.print("V, Current: ");
  Serial.print(reading.current, 2); // Print current with 2 decimal places
  Serial.print("A, Date and Time: ");
  Serial.println(reading.dateTime.toString("YYYY-MM-DDThh:mm:ss"));
   Serial.print(".");
  if (reading.milliseconds < 10) {
    Serial.print("00");
  } else if (reading.milliseconds < 100) {
    Serial.print("0");
  }
  Serial.print(reading.milliseconds);

  Serial.println("Z");

}
```

Figure 8. printreading code snippet

sendDataToThingspeak(): The core of the code lies in this function, which handles the transmission of data to the ThingSpeak cloud platform. It utilizes the DFRobot_SIM7000 library (DFRobot,

2018)to establish a network connection, construct HTTP GET requests containing the measurement data and send the requests to ThingSpeak. An HTTP GET request is a method the client uses to request data from a specified resource on a server. This is one of the most common HTTP methods, primarily used to retrieve data from a web server. The structure of an HTTP GET request includes the HTTP method (GET), the URL of the resource, and optional headers. The success or failure of data transmission is monitored, with appropriate actions taken in each scenario (Figure 8).

```
void sendDataToThingspeak() {
  for (int currentIndex = 0; currentIndex <= count; currentIndex++) {
    if (sim7000.openNetwork(DFRobot_SIM7000::eTCP, "api.thingspeak.com", 80) != 0) { // Check port of ThingSpeak
      Serial.println("Failed to open network connection. Retrying...");
      delay(5000); // Wait for a few seconds before retrying
      continue; // Retry opening the network connection
    }

    // Get the date and time as a formatted string
   String url = "GET /update?api_key=" + String(THINGSPEAK_API_KEY) +    //url
              "&field3=" + String(readings[currentIndex].voltage, 2) +
              "&field4=" + String(readings[currentIndex].current, 2) +
              "&field5=" + readings[currentIndex].dateTime.toString("YYYY-MM-DDThh:mm:ss") +
              "." + readings[currentIndex].milliseconds +
              "Z HTTP/1.1\r\nHost: api.thingspeak.com\r\nConnection: close\r\n\r\n";

    Serial.print("Sending data to ThingSpeak: ");
    Serial.println(url);

    if (sim7000.send((char*)url.c_str())) {
      // Data sent successfully
      sim7000.closeNetwork();
    } else {
      Serial.println("Failed to send data. Retrying...");
      delay(2000); // Wait for a few seconds before retrying
      sim7000.closeNetwork(); // Close the network connection
      currentIndex--; // Retry the current index
    }

    delay(10000); // Wait for 15 seconds before sending the next set of data
  }
}
```

Figure 9. Code snippet for data transmission to Thingspeak

timerISR(): The timer interrupt service routine (ISR) plays a crucial role in triggering periodic readings. By using the millis() function, we ensure that voltage and current values are consistently captured at 2-millisecond intervals. This approach aligns with the configured timer settings and helps maintain the accuracy and reliability of the measurement system. The Arduino code is structured and modular, encapsulating specific functionalities within well-defined functions. This promotes readability, ease of debugging, and efficient execution, all crucial aspects of a successful implementation. The methodology's reliance on Arduino programming showcases the flexibility and versatility of this platform in facilitating IoT projects, particularly those involving NB-IoT technology and cloud integration.

```
void loop() {
  unsigned long startMillis = millis();  // Start time of the loop

  // Measure the total time taken for sampling
  unsigned long samplingStartTime = micros();
  for (int i = 0; i < samples; i++) {
    unsigned long iterationStartTime = micros();

    readings[readingCount] = takeReading();
    readingCount++;

    // Wait until the next 2ms interval
    while (micros() - iterationStartTime < 2000);
  }
  unsigned long samplingEndTime = micros();
  Serial.print("Total sampling time: ");
  Serial.print((samplingEndTime - samplingStartTime) / 1000.0);
  Serial.println(" ms");

  delay(5000); // Delay between iterations
}
```

Figure 10. millis code snippet

4 MEASUREMENT AND ANALYSIS

4.1 Explanation of the voltage and current measurement process

The voltage and current measurement process in this project involves a systematic approach to accurately capture electrical parameters from an AC network (Figure 4) using sensors connected to an Arduino Mega. In this project, the term "AC network" refers to the standard 230V AC mains electricity supply commonly found in residential and commercial buildings. The measurements are taken from this 230V AC network to monitor electrical parameters such as voltage and current. The integration of Narrowband Internet of Things (NB-IoT) technology facilitates the transmission of these measurements to a cloud platform for further analysis. The following section provides a detailed explanation of the voltage and current measurement process.

4.1.1 Sensor Configuration

In this project, using two crucial sensors, a voltmeter, and an ammeter, connected to the Arduino Mega's analog pins A0 and A1, respectively, is paramount for capturing real-time voltage and current values from an AC network.

Voltmeter (SI-9002): The SI-9002 (instruments) voltmeter (Figure 11) is used to measure the electrical potential difference across a circuit. This device provides insight into the voltage levels at specific points within the system. The SI-9002 voltmeter is connected in parallel to the AC mains, specifically between the phase and neutral wires, to accurately measure the voltage. The voltmeter output is then connected to a level shifter to ensure compatibility with the Arduino's input range, before being fed into analog pin A0 of the Arduino Mega. Output Range: 0-5V (after level shifter).

Figure 11. Voltmeter Model SI-9002

Bandwidth	DC to 25MHz (-3dB)
Attenuation Ratio	1:20/200
Accuracy	±2%
Rise Time	14ns
Input Impedance	4MΩ//5.5pF each side to ground
Input Voltage	
-Category	CAT III
-Differential Range	±140V(DC+AC Peak) and 140Vrms @1/20
	±1400V(DC+AC Peak) and 1000Vrms @1/200
- Common Mode Range	±1400V(DC+AC Peak) and 1000Vrms @1/20 & 1/200
- Absolute Max. Voltage (Differential or Common Mode)	±1400V(DC+AC Peak) and 1000Vrms @1/20 & 1/200
Output Voltage	
- Swing	±7V (into 50kΩ load)
- Offset (typical)	<±5mV
- Noise (typical)	0.7mVrms
- Source Impedance (typical)	50Ω (for using 1MΩ input system oscilloscope)
CMRR (typical)	-86dB @50Hz, -60dB @20kHZ
Ambient Operating Temperature	-10 to 40℃
Ambient Storage Temperature	-30 to 70℃
Ambient Operating Humidity	Up to 85% RH
Ambient Storage Humidity	Up to 85% RH
Power Requirements*	
- Standard	4xAA cells
- Options	Power leads, Mains adaptor* (6VDC/60mA or regulated 9VDC/40mA), USB power cord
Length of BNC Cable	95cm
Length of Input Leads	45cm
Weight	400gms (probe and PVC jacket)
Dimensions (LxWxH)	170mm x 63mm x 21mm

Figure 12. SI-9002 Specifications

Ammeter (Fluke i30s): The Fluke i30s (Fluke) An ammeter (Figure 12), a clamp meter, measures the current passing through the circuit. It clamps around the neutral wire of the AC network to capture the flow of electric charge. The output from the ammeter is also routed through a level shifter to match the Arduino's input requirements and connected to analog pin A1 of the Arduino Mega. Output Range: 0-5V (after level shifter).

Figure 13. Ammeter Fluke i30s

General specifications	
Maximum conductor size	19 mm (.748 in) diameter
Output connection	Safety BNC connector, supplied with safety 4 mm (.157 in) adapter
Output zero	Manual adjust via thumbwheel
Cable length	2 m (6.56 ft)
Operating temperature range	0 °C to +50 °C (32 °F to 122 °F)
Storage temperature range (with battery removed)	-20 °C to +85 °C (-4 °F to 185 °F)
Operating humidity	15 % to 85 % (non-condensing)
Dimensions (HxWxD)	183 mm x 71 mm x 25 mm (7.2 in x 2.8 in x 1 in)
Weight	250 g (.55 lb)
Electrical specifications	
Specified current range	30 mA to 30 A DC, 30 mA to 20 A AC rms
Usable current range	5 mA to 30 A DC, 30 mA to 20 A AC rms
Crest Factor	1.4
Output sensitivity	100 mV/A
Accuracy (at +25 °C)	DC ± 1 % of reading ± 2 mA AC ± 0.5 dB of reading ± 2 mA
Resolution	± 1 mA
Load impedence	> 100 k Ohms ≤ 100 pF
Conductor position sensitivity	± 1 % relative to center reading
Frequency range	DC to 100 kHz (0.5 dB)
Phase shift below 1 kHz	< 2 degrees
Temperature coefficient	± 0.01 % of reading/°C
Power supply	9 V Alkaline, IEC 6LR61, 30 hours, low battery indicator
Working voltage (see safety standards)	300 V AC rms or DC

Figure 14. Fluke i30s Specifications

4.1.2 Analog-to-Digital Conversion

The Arduino Mega, equipped with analog-to-digital converters (ADCs), reads analog signals from the sensors. The analog readings, obtained as discrete values, are then converted into digital values ranging from 0 to 1023. This conversion enables the Arduino to interpret and process the voltage

and current measurements in a format suitable for further analysis (Pelgrom, 2013).

Voltage Range of Arduino Mega ADC

The Arduino Mega's ADC has a default voltage range of 0 to 5V. This means that any analog input signal within this range can be read and converted to a digital value. The ADC provides a 10-bit resolution, which divides the input range into 1024 discrete steps (0 to 1023). Each step represents a voltage increment of approximately 4.88 mV (5V/1024 steps).

Reference Voltage

The choice of reference voltage (V_ref) is crucial for accurate measurements. The default reference voltage for the Arduino Mega's ADC is 5V, but it can be adjusted using the analogReference() function to other predefined values such as 1.1V (internal) or the AREF pin for an external reference voltage.

For this project, the default reference voltage of 5V is used, which corresponds well with the level-shifted signals from the voltmeter and ammeter.

Relationship between Sensor Output and ADC

1. Voltmeter (SI-9002) Output:
    - Sensor Output: The SI-9002 voltmeter outputs a voltage that represents the measured AC voltage. This output needs to be within the 0-5V range for the Arduino to read it correctly.
    - Level Shifter: The output from the SI-9002 is passed through a level shifter to scale down the high AC voltage to a safe 0-5V range.
    - ADC Input: The scaled voltage is connected to analog pin A0. The ADC converts this signal into a digital value between 0 and 1023.
    - Calculation: The actual AC voltage can be calculated using the formula:

$$V_{AC} = \frac{V_{ADC}}{1023} \cdot 5V \cdot Scaling\ Factor$$

Where V_{ADC} is the digital value read by the ADC.

2. Ammeter (Fluke i30s) Output:
    - Sensor Output: The Fluke i30s clamp meter outputs a voltage proportional to the measured current.
    - Level Shifter: This output is also routed through a level shifter to ensure the signal is within the 0-5V range.
    - ADC Input: The scaled current signal is connected to analog pin A1. The ADC then digitizes this signal.
    - Calculation: The actual current can be determined using the formula:

$$V_{AC} = \frac{I_{ADC}}{1023} \cdot 5V \cdot Scaling\ Factor$$

Where I_{ADC} is the digital value from the ADC.

By understanding these relationships, the Arduino Mega can accurately interpret the voltage and current signals from the sensors, enabling precise monitoring and analysis of the AC network.

4.1.3 Calibration and Accuracy

Calibration processes may be implemented to ensure accuracy in voltage and current measurements. Calibration involves comparing the sensor readings with known reference values and adjusting the system to minimize any discrepancies. This step is crucial for compensating for potential sensor inaccuracies and enhancing the reliability of the measurements.

Calibration Process

In this project, the calibration process primarily focused on setting the appropriate attenuation ratio for the voltmeter. The following outlines the calibration steps taken:

Attenuation Ratio Setting for the Voltmeter

1. Attenuation Ratio Configuration:
    - The SI-9002 voltmeter was configured with an attenuation ratio of 1/200. This setting ensures that the high voltage from the AC network is scaled down to a safe level that the Arduino Mega can read.
    - With an attenuation ratio of 1/200, an input voltage of 200V would be scaled down to 1V at the voltmeter output.

Verification and Adjustment

To verify the accuracy of the attenuation ratio setting and the overall measurement system, the following steps were taken:

1. Known Reference Voltage:
    - A known reference voltage was applied to the input of the SI-9002 voltmeter.
    - The expected output voltage was calculated based on the 1/200 attenuation ratio.
2. Measurement and Comparison:
    - The Arduino Mega read the output voltage from the voltmeter.
    - This reading was compared to the expected output voltage.

4.1.4 Timestamping Using the RTC (DS3231) for Accurate Timestamping

The DS3231 RTC module is integrated into the system to provide accurate timekeeping. The following steps outline how the RTC is used to timestamp each measurement:

1. RTC Initialization:
    - The DS3231 RTC module is initialized at the start of the program.

- The I2C interface is used to communicate with the RTC module, with connections made as follows:
    - SDA (RTC) → A4 (Arduino Mega)
    - SCL (RTC) → A5 (Arduino Mega)

2. Retrieving the Current Time:
    - Before each measurement is taken, the current date and time are retrieved from the RTC module.
    - The RTC provides a timestamp in a human-readable format (e.g., YYYY-MM-DD HH:MM:SS).
3. Associating Timestamps with Measurements:
    - Each voltage and current measurement are recorded along with the timestamp from the RTC.
    - This ensures that each data point is accurately time-stamped, providing precise temporal context.

4.1.5 Data Storage in Struct

The acquired voltage, current, and timestamp values are organized and stored in a structured format using a custom-defined Reading struct. This struct serves as a container for storing the measured values, allowing for efficient data organization and retrieval. By structuring the data in this manner, the system can easily access and manipulate individual measurements during subsequent processing and transmission steps. This structured approach to data storage streamlines the overall data management process, enhancing system efficiency and performance. (Tutorialspoint)

4.1.6 Periodic Readings with millis()

To ensure accurate and reliable measurements of a 50Hz AC signal, it is essential to sample the waveform at a sufficient rate to capture its details. The Nyquist theorem states that to reconstruct a signal accurately, it must be sampled at least twice its highest frequency. For a 50Hz signal, this would imply a minimum sampling rate of 100Hz. However, a higher sampling rate is preferred to obtain more precise measurements and capture the waveform's nuances.

Understanding the Nyquist Theorem

The Nyquist theorem (Por 2019, 5) states that the minimum sampling rate f_S required to accurately reconstruct a signal with maximum frequency $fmax$ is given by:

$$f_s \geq * f_{max}$$

For a 50Hz AC signal:

$$f_{\max} = 50Hz$$
$$f_S \geq \ 2 * 50Hz = 100Hz$$

This means the signal should be sampled at least 100 times per second, or every 10 milliseconds. Sampling the signal at 2ms intervals corresponds to a sampling frequency of:

$$f_S = \frac{1}{0.002sec} = 500Hz$$

Calculating the Sampling Rate

Signal Frequency: 50Hz

Nyquist Rate: At least 100Hz (minimum sampling rate)

chosen Sampling Rate: 500Hz (2ms interval, providing 10 samples per cycle of the 50Hz signal)

By sampling at a rate of 500Hz (every 2ms), the system captures 10 samples per cycle of the 50Hz signal. This higher sampling rate allows for a more detailed and accurate representation of the AC waveform, improving the quality of the analysis and enabling the detection of subtle changes in the signal.

Why 2ms is chosen

Adequate Sampling Rate: The chosen 2ms interval provides a sampling rate of 500Hz, which is significantly higher than the minimum required Nyquist rate of 100Hz. This ensures a more detailed representation of the 50Hz signal.

Data Quality: A higher sampling rate reduces aliasing and improves the accuracy of the measurements.

Real-time Monitoring: The 2ms interval allows the system to continuously monitor the electrical parameters in near real-time, providing timely data for analysis and decision-making.

Improved Signal Reconstruction: A higher sampling rate than the minimum requirement results in better resolution and accuracy in reconstructing the AC waveform. This allows for a more detailed analysis of the voltage and current signals.

Capturing Harmonics: Higher sampling rates enable the capture of higher frequency components (harmonics) present in the AC signal, which are critical for analyzing power quality and detecting anomalies.

4.1.7 Real-time Monitoring through Serial Output

For real-time monitoring and debugging purposes, the system utilizes the Arduino IDE`s Serial Monitor (Söderby 2024) to output the acquired voltage, current, and timestamp data. By printing this information to the Serial Monitor, users can visually inspect the data in real-time, verify the accuracy of measurements, and ensure the proper functioning of the system. This real-time feedback mechanism is essential for troubleshooting potential issues, calibrating sensor readings, and validating system performance during operation. (Badamasi 2014, 1-4.)

4.1.8 Integration with NB-IoT for Cloud Transmission

Following the data acquisition process, the measurements are transmitted to the ThingSpeak cloud platform using NB-IoT technology. The sendDataToThingspeak() function establishes a network connection, constructs HTTP GET requests, and sends the data to ThingSpeak for storage and analysis. In summary, the voltage and current measurement process is a multistep procedure involving sensor configuration, analog-to-digital conversion, calibration, timestamping, and periodic readings

orchestrated by a timer interrupt. The integration of NB-IoT technology ensures seamless transmission of these measurements to a cloud platform, enabling remote monitoring and analysis. This process forms the foundation for precise and efficient electrical parameter monitoring in IoT applications. (Pasha 2016, 19-23.)

Figure 15. Data represented in Thingspeak Dashboard

4.2 Discussion of challenges encountered during measurement

The implementation of the voltage and current measurement system using Arduino and NB-IoT technology faced several challenges, each of which required careful consideration and innovative solutions to ensure accurate and reliable measurements. One credible challenge was the Arduino's inability to read negative values directly, prompting the utilization of a voltage level shifter to address this limitation.

4.2.1 Challenge of Arduino`s Limited Capability for Negative Values

Arduino Mega, renowned for its versatility in numerous electronic projects, comes with built-in analog-to-digital converters (ADCs) that facilitate the conversion of analog signals to digital values. However, its default configuration limits ADC readings to voltages ranging from 0 to 5 volts. This range suffices for many applications involving sensors, actuators, and other analog components. Yet, when confronted with alternating current (AC) signals, particularly those oscillating around a zero-voltage reference point, Arduino encounters a significant hurdle.

AC signals exhibit a waveform that alternates between positive and negative values as it oscillates around the zero-voltage axis. While these swings contain crucial information, Arduino's ADCs can

only interpret positive voltages. This inherent limitation arises due to the structure of the ADCs and their reference points. The ADCs within Arduino operate by comparing the input voltage against a reference voltage, typically 5 volts for Arduino Uno and Mega boards. Consequently, any voltage below the reference level is registered as 0, effectively disregarding the negative portion of the AC waveform. (Fransiska 2013, 226-229.)

This inability to read negative values poses a fundamental challenge when attempting to measure AC signals accurately. Since the negative swings of the waveform are ignored, vital information regarding the signal's magnitude and characteristics is lost. Consequently, the captured data may contain inaccuracies, leading to erroneous interpretations and analyses.

In essence, while Arduino Mega excels in numerous applications, its default ADC configuration presents limitations when confronted with AC signals. Overcoming this challenge requires innovative solutions, such as voltage level shifting, to ensure accurate measurement and reliable data capture in projects involving AC circuits and systems.

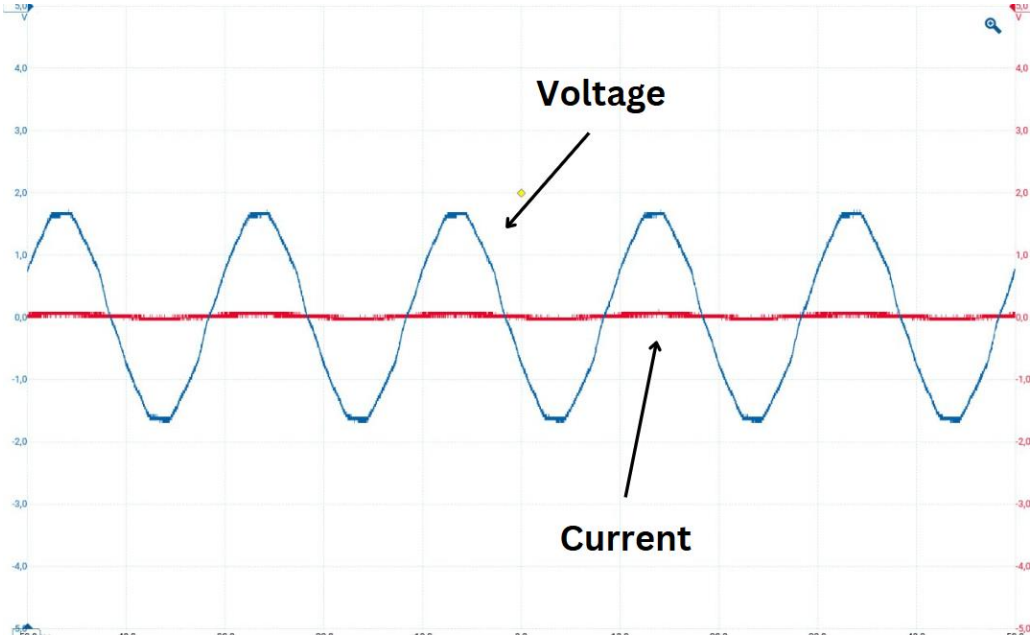

Figure 16. Original Voltage and Current Signals of 50Hz

4.2.2 Solution and Integration of a Voltage Level Shifter

To address the challenge posed by negative values in the AC waveform, a voltage level shifter (Figure 17) was integrated into the circuit. The voltage level shifter plays a crucial role in shifting the entire AC signal upwards by a certain voltage offset, effectively repositioning the waveform within the Arduino's readable range. By shifting the waveform upwards, the voltage level shifter ensures that the entire AC signal falls within the 0 to 5-volt range that the Arduino can accurately interpret.

The voltage level shifter circuit includes components such as diodes and capacitors. Specifically, a 47µF capacitor and diode are utilized in the circuit design. The capacitor helps stabilize the voltage

levels, while the diode assists in rectifying the AC signal, allowing only the positive component to pass through.

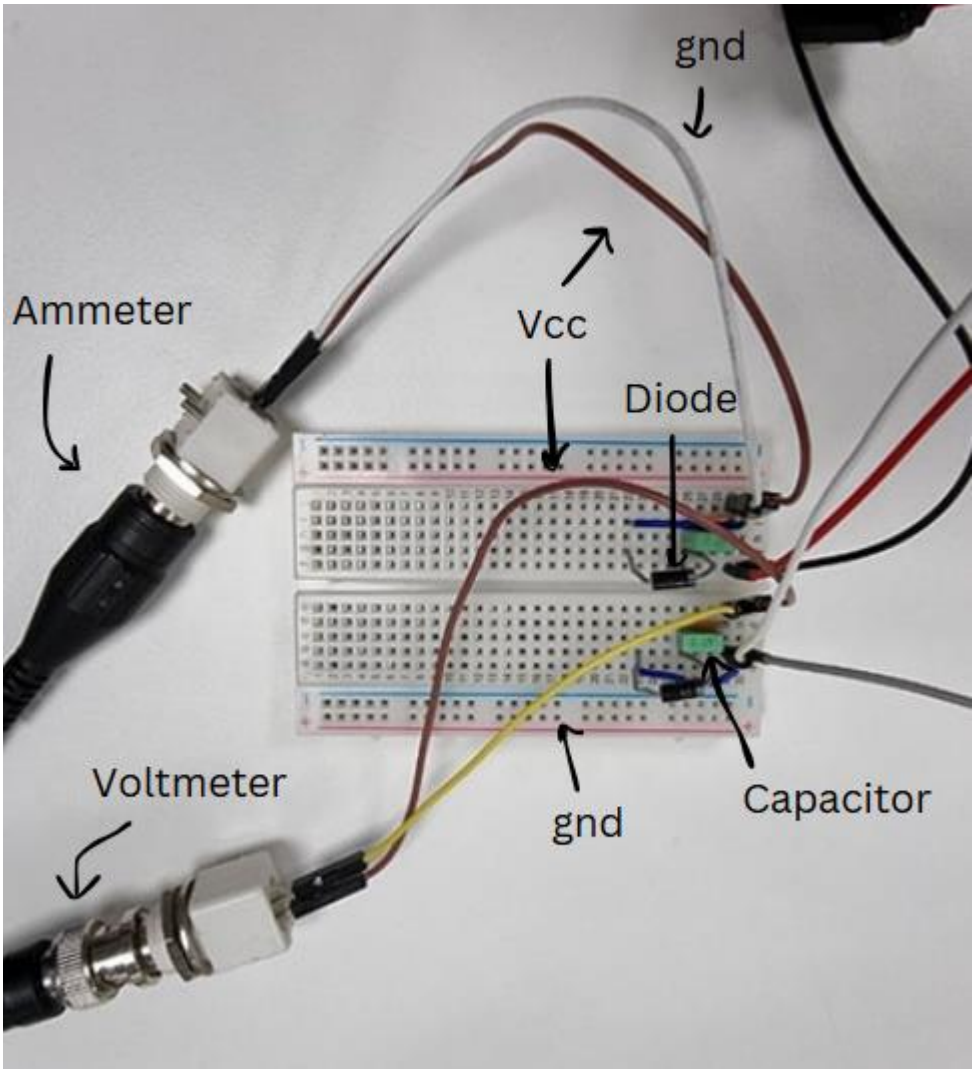

Figure 17. Voltage Level Shifter

A voltage level shifter is an electronic circuit (Figure 18) designed to shift the level of an input signal to a different voltage range, allowing compatibility with devices that operate at different voltage levels. In the context of AC waveform measurement using an Arduino, the voltage level shifter adjusts the entire AC signal so that it falls within the 0 to 5-volt range that the Arduino's ADC can read.

Working Principle of the Voltage Level Shifter

To understand the voltage level shifter, let's delve into the core components and their roles:

Diodes: A Zener diode allows current to flow forward like a regular diode but also permits current to flow in the reverse direction when the voltage is above a certain value (the Zener breakdown voltage). This characteristic is used to create a fixed DC offset.

Capacitors: Capacitors, such as the 47µF capacitor mentioned, are used for filtering and stabilizing voltage levels.

By integrating a voltage level shifter into the circuit, the system can accurately capture both positive and negative swings of the AC waveform (Figure 19), enabling more precise voltage and current measurements. This solution enhances the reliability and accuracy of the measurement system, ensuring that no information is lost during the analog-to-digital conversion process. (Horowitz 1989, 658)

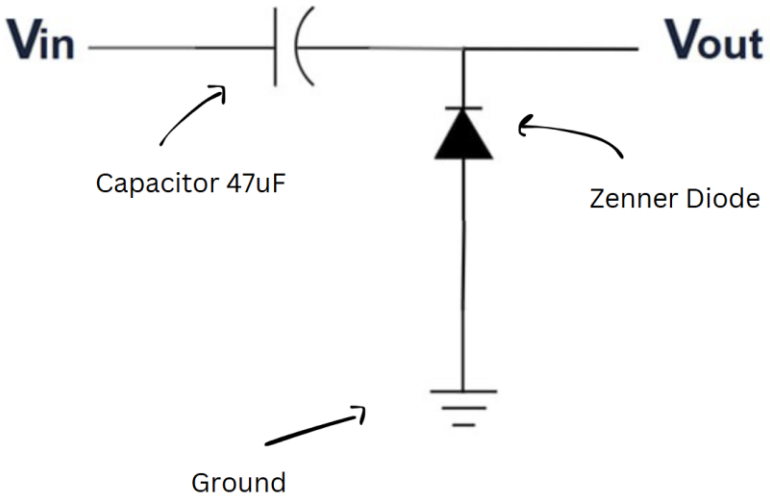

Figure 18. Voltage level Shifter Diagram

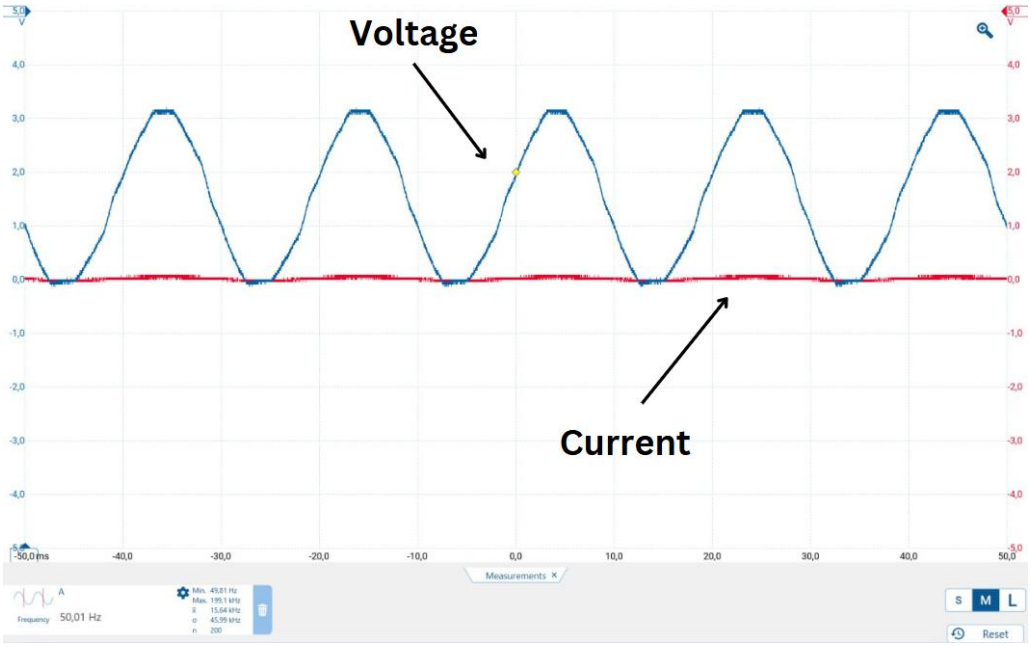

Figure 19. Shifted Voltage and Current Signals of 50Hz

4.2.3 Additional Considerations and Implications

While the voltage level shifter effectively addresses the issue of negative values, it introduces an additional layer of complexity to the circuit design. The selection of an appropriate voltage offset becomes critical, as it determines the extent to which the signal is shifted and, consequently, affects the accuracy of measurements. Furthermore, the integration of external components, such as the voltage level shifter, necessitates careful calibration to align the measurements with the actual AC signal. Calibration procedures were implemented to fine-tune the system and ensure that the readings accurately reflected the true electrical parameters.

4.2.4 Future Considerations and Improvements

In future iterations of the project, the exploration of alternative solutions to handle negative values without the need for external components could be considered. Additionally, advancements in microcontroller technology or the incorporation of dedicated signal conditioning circuits might provide more seamless solutions to address this challenge. In conclusion, the challenge posed by Arduino's inability to read negative values in AC signals was successfully mitigated by the strategic integration of a voltage level shifter. This discussion highlights the importance of identifying and overcoming technical limitations during the implementation of IoT projects, underscoring the adaptability and problem-solving skills inherent in the development process.

4.3 Presentation of collected data samples and their significance

The successful implementation of the voltage and current measurement system using Arduino and NB-IoT technology has yielded a dataset rich in electrical parameter readings. This section focuses on presenting selected data samples and elucidating their significance within the context of the project's objectives.

4.3.1 Data Sample Representation

The collected data samples (Figure 20) encompass voltage, current, and corresponding timestamps, each stored in the structed reading format. A subset of this dataset is presented for analysis and interpretation.

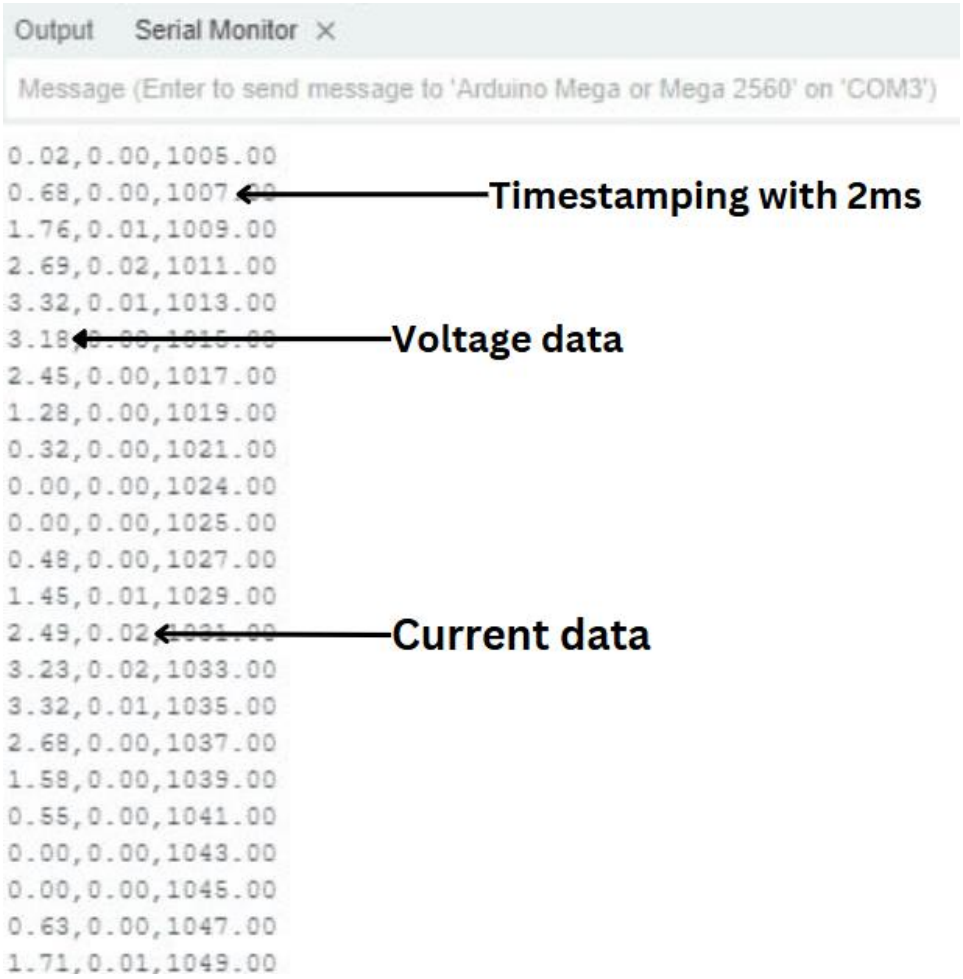

Figure 20. Example of collected data in Arduino serial monitor

4.3.2 Analysis of Voltage and Current Trends

The dataset obtained from the voltage and current measurements enables a comprehensive analysis of trends over time. By generating plots and graphs from the collected data, it becomes possible to visually represent fluctuations, peaks, and troughs in both voltage and current readings. These visualizations serve as powerful tools for identifying patterns that may correlate with specific electrical events or operational conditions. For example, recurring spikes or dips in voltage may indicate irregularities in the power supply, while fluctuations in current could signify changes in electrical load. By analyzing these trends, it becomes easier to detect anomalies, anticipate potential issues, and optimize system performance.

4.3.3 Temporal Dynamics

The inclusion of timestamps in each data sample facilitates the analysis of temporal dynamics. Temporal trends are crucial for understanding how voltage and current parameters evolve over different periods, such as hourly, daily, or weekly cycles. This temporal granularity provides insights into the system's behavior and can reveal patterns related to usage patterns or external factors.

4.3.4 Significance for IoT Applications

The presentation and analysis of collected data samples hold significant implications for IoT applications. Understanding electrical parameters in real-time empowers stakeholders to make informed decisions, implement predictive maintenance strategies, and optimize energy usage. By leveraging cloud connectivity, the data collected from voltage and current measurements can be transmitted and analyzed remotely, facilitating proactive management of electrical systems. For example, anomalies detected in voltage or current trends can trigger automatic alerts, prompting timely intervention to prevent potential downtime or equipment failures. Furthermore, insights gleaned from data analysis can inform long-term planning, enabling organizations to optimize resource allocation, improve operational efficiency, and enhance overall system reliability. The significance of analyzing voltage and current data lies in its potential to drive actionable insights and facilitate continuous improvement in IoT-enabled monitoring systems. (Morello 2017, 7828-7837.)

5 CLOUD INTEGRATION AND DATA VISUALIZATION

This section elucidates the integration of the voltage and current measurement system with the Thingspeak platform, highlighting the nuances of data transmission from Arduino to Thingspeak. Additionally, it provides a detailed overview of the Thingspeak platform, explores the generated data visualizations, and includes a comparative analysis between Thingspeak and Azure for data processing and analysis.

5.1 Detailed Overview of Thingspeak Platform

Thingspeak, chosen as the cloud platform for this project, offers a comprehensive environment for IoT applications. Its user-friendly interface, coupled with robust capabilities, makes it an ideal choice for data storage, visualization, and analysis.

Key features of Thingspeak

1. Real-time Data updating
   - Description: Thingspeak supports real-time data streaming, which enables users to receive, visualize, and analyze data as it is collected.
   - Capabilities:
     - Instant data updates with minimal latency.
     - Live dashboards to monitor ongoing processes.
     - Customizable visual widgets to display real-time data.
2. Chanel Customization
   - Description: Channels in ThingSpeak are used to store and organize data streams. Each channel can hold multiple data fields.
   - Capabilities:
     - Up to 8 data fields per channel.
     - Metadata fields for descriptive information.
     - Public or private channel settings for data sharing.
     - Advanced data categorization and tagging options.

3. Support for Various IoT Devices
   - Description: ThingSpeak is compatible with many IoT devices and platforms, making it versatile and adaptable.
   - Capabilities:
     - Integration with popular hardware like Arduino, Raspberry Pi, and ESP8266/ESP32.
     - Compatibility with various communication protocols such as HTTP, MQTT, and CoAP.
     - API support for custom device integration.
4. Data Visualization
   - Description: ThingSpeak provides robust tools for visualizing data through customizable charts and plots.
   - Capabilities:
     - Time-series plots for historical data analysis.
     - Real-time graphs for live data monitoring.
     - Options for line charts, bar graphs, pie charts, and more.
     - MATLAB integration for advanced data analysis and custom visualizations.
5. Data Analysis and Alerts
   - Description: ThingSpeak offers built-in MATLAB analytics and the ability to trigger alerts based on data thresholds.
   - Capabilities:
     - Perform complex calculations and data transformations using MATLAB code.
     - Create alerts and notifications via email, SMS, or third-party services (e.g., IFTTT) based on predefined conditions.
     - Implement machine learning algorithms for predictive analysis.
6. Data Storage and Export
   - Description: ThingSpeak provides reliable data storage solutions and options for data export.
   - Capabilities:
     - Long-term data storage for historical analysis.
     - Data export in CSV or JSON format for external use.
     - RESTful API access for data retrieval and manipulation.

ThingSpeak offers a comprehensive and adaptable environment for IoT applications, particularly for voltage/current monitoring systems. Its real-time data capabilities, extensive customization options, and support for various IoT devices make it a robust choice for managing and analyzing IoT data efficiently. (Maureira 2011, 1-4.)

5.2 Data Transmission from Arduino to Thingspeak

Ensuring seamless data transmission from the Arduino microcontroller to the Thingspeak platform is crucial for the project's success. This subsection outlines the protocols, APIs, and communication mechanisms employed for data transfer. Key aspects of data transmission include:

Protocol Selection: The project selects appropriate communication protocols, such as HTTP, to facilitate the transfer of data from the Arduino to Thingspeak.

API Integration

ThingSpeak provides a robust API that enables easy data integration. In this project, the API write key is used to authenticate and send data to specific channels on ThingSpeak. The API write key is a unique identifier that ensures data is sent to the correct channel. The following steps outline the process of sending data to ThingSpeak using the API:

Prepare Data: Collect the voltage and current measurements from the sensors connected to the Arduino.

Construct HTTP GET Request: Format the measurements into an HTTP GET request string, including the API write key and the data fields.

Send Request: Use the Arduino's network capabilities to send the HTTP GET request to the ThingSpeak server.
Verify Response: Ensure that ThingSpeak responds with a success code, indicating that the data has been successfully received and stored.

Reliability and Efficiency: Emphasis is placed on the reliability and efficiency of the data transmission process, minimizing latency, and ensuring that measurements are consistently and accurately updated on the Thingspeak platform. (Kandimalla 2017, 3.)

5.3 Presentation and Implementation of Data Visualizations

Thingspeak's data visualization capabilities enable the creation of dynamic and insightful visualizations based on the transmitted data. Representative visualizations, including line charts, scatter plots, and gauges, showcase voltage and current trends over time. Interpretations of these visualizations shed light on patterns, anomalies, and the overall behavior of electrical parameters, providing a user-friendly interface for monitoring. Key points include:

Visualization Types: Various visualization types are explored, each offering unique insights into the data. Line charts are ideal for tracking trends over time, while scatter plots can reveal correlations between variables.

Anomaly Detection: Visualizations are analyzed for anomalies or irregularities in voltage and current measurements, enabling proactive identification of potential issues or abnormalities in the system.

5.4 Comparison of Thingspeak and Azure

In the realm of cloud platforms for IoT applications, both Thingspeak (mathworks) and Azure (Azure 2016) offer robust solutions with distinct features and capabilities. A comparative analysis between these platforms illuminates their respective strengths and weaknesses, guiding the selection process for specific project requirements.

Ease of Integration:

Thingspeak: Known for its user-friendly interface and simplicity, Thingspeak provides straightforward integration with Arduino and other IoT devices. Its RESTful API and extensive documentation make it easy for developers to set up data streams and visualizations. Azure: Azure offers a comprehensive suite of IoT services but requires a steeper learning curve for integration. While it provides powerful tools and APIs, setting up and configuring Azure services may be more complex compared to Thingspeak.

Scalability:

Thingspeak: Designed primarily for smaller-scale IoT projects, Thingspeak may face limitations in handling large volumes of data or high-throughput applications. It's suitable for prototyping and smaller deployments but may not scale efficiently for enterprise-level solutions. Azure: Azure excels in scalability, offering a wide range of services capable of handling massive amounts of data and supporting enterprise-grade IoT deployments. Its scalability makes it ideal for projects with growing data needs and complex infrastructures.

Data Storage Capabilities:

Thingspeak: While Thingspeak provides adequate data storage for many IoT applications, its storage capabilities may be limited compared to Azure. For projects with extensive data retention requirements or advanced analytics, Thingspeak's storage options may fall short. Azure: Azure offers a plethora of storage options, including Blob storage, Table storage, and Cosmos DB, providing flexibility and scalability for storing IoT data. Azure's storage solutions are highly customizable and can accommodate various data types and volumes.

Real-time Processing:

Thingspeak: Thingspeak supports real-time data updating and processing, making it suitable for applications requiring immediate insights or alerts based on incoming data. However, its real-time processing capabilities may be more basic compared to Azure. Azure: Azure provides advanced real-time processing capabilities through services like Azure Stream Analytics and Azure Functions. These services enable complex event processing, machine learning inference, and real-time analytics, empowering developers to derive actionable insights from streaming data.

Accessibility of Analytical Tools:
Thingspeak: Thingspeak offers built-in visualization tools for creating basic charts and graphs to visualize IoT data. While these tools are easy to use, they may lack the sophistication and customization options available in Azure. Azure: Azure provides a comprehensive suite of analytical tools, including Power BI, Azure Machine Learning, and Azure Synapse Analytics. These tools enable advanced data analysis, predictive modeling, and business intelligence, offering unparalleled insights into IoT data. (Toutsop 2021, 413-420.)

5.5 Considerations for Platform Selection

The evaluation of Thingspeak and Azure encompasses not only their current features but also potential scalability and adaptability for future expansions of the IoT system. Considerations are given to the specific requirements of the project, such as the volume of data, real-time processing needs, and compatibility with additional IoT devices.

5.6 Practical Insights and Recommendations

Based on a comparison of Thingspeak and Azure, the choice of Thingspeak for this project was due to several factors:

Ease of Integration: Thingspeak's simplicity and compatibility with Arduino make it easier to set up and integrate quickly, which is consistent with the project's goals of rapid prototyping and experimentation.

Scalability: Although Azure provides excellent scalability for enterprise-level deployments, Thingspeak's scalability was considered sufficient to scale a project that was focused on smaller-scale monitoring systems.

Real-time processing: Thingspeak's support for real-time data updates met the project's requirements for instant information and responsiveness.

Cost-effectiveness: For small projects or projects with a limited budget, Thingspeak's free tier and simple pricing structure provide a cost-effective solution without compromising basic functions.

In conclusion, while Azure may offer more advanced capabilities and scalability for large-scale IoT deployments, Thingspeak has become the preferred choice for this project due to its ease of use, suitability for smaller scale applications, and cost-effectiveness. However, the choice of a cloud platform should always be determined by the specific requirements and goals of each IoT project.

In addition to the factors mentioned, another important aspect influencing the choice of Thingspeak for this project is its integration with MATLAB. Integration with MATLAB is important to our project because of its potential for future data analysis and modeling. Using MATLAB's powerful analytical capabilities, we can perform in-depth analysis of the collected data directly on the Thingspeak

platform. This integration simplifies the workflow and expands the project's capabilities for advanced data processing, algorithm development and visualization. Thus, the smooth integration of Thingspeak with MATLAB increases the platform's value by another level, making it an even more attractive choice for our IoT project. (Higham 2016.)

6 CONCLUSION AND DISCUSSION

6.1 Summary of Findings

This thesis presents a comprehensive approach to accurately measure voltage and current in an AC network using an Arduino Mega, and subsequently transmit the data to the ThingSpeak cloud platform via NB-IoT technology. The primary objectives of this project were to design a reliable system for real-time monitoring of electrical parameters and to ensure seamless data transmission for remote analysis. The following summarizes the key findings and contributions of this work:

1. System Design and Implementation:
    - Utilization of an Arduino Mega equipped with analog-to-digital converters (ADCs) to read analog signals from a voltmeter (SI-9002) and an ammeter (Fluke i30s), connected to analog pins A0 and A1, respectively.
    - Implementation of a voltage level shifter using a Zener diode and capacitor to ensure the AC signal falls within the Arduino's readable range, addressing the challenge of negative values in the AC waveform.
2. Accurate Data Acquisition:
    - The system was designed to capture voltage and current readings at 2-millisecond intervals using a timer interrupt service routine (ISR). This sampling rate was justified based on the need to accurately measure a 50Hz AC signal.
    - The readings were timestamped using the millis() function, providing a temporal context for tracking variations in electrical parameters over time.
3. Calibration and Signal Processing:
    - Calibration processes were discussed to minimize sensor inaccuracies and enhance measurement reliability. The system was set with a voltmeter attenuation ratio of 1/200 to ensure accurate voltage readings.
4. Cloud Integration and Data Transmission:
    - Integration with ThingSpeak's API allowed for the seamless transmission of voltage, current, and timestamp data to the cloud platform. The sendDataToThingspeak() function was employed to establish a network connection and send HTTP GET requests to ThingSpeak for data storage and analysis.
    - A detailed comparison between ThingSpeak and Azure highlighted the strengths and limitations of each platform, with ThingSpeak being user-friendly and suitable for smaller-scale projects, while Azure offered advanced capabilities and scalability for larger deployments.
5. Visualization and Analysis:

- Real-time data updating and channel customization in ThingSpeak facilitated efficient data management and visualization. The platform's compatibility with various IoT devices enabled versatile hardware configurations and robust data handling.

6.2 Recommendations for Future Work

While the project successfully achieved its primary objectives, several areas offer potential for further enhancement and exploration:

1. Enhanced Signal Processing:
   - Incorporate advanced digital filtering techniques, such as FIR and IIR filters, to reduce noise and artifacts in the measured signals, ensuring cleaner and more accurate data.
2. Expanded Cloud Capabilities:
   - Explore integration with additional cloud platforms like Azure for projects requiring higher scalability and advanced analytical tools. Leveraging services like Azure Stream Analytics and Azure Machine Learning can provide deeper insights through real-time analytics and predictive modeling.
3. Data Security and Privacy:
   - Address data security and privacy concerns by implementing robust encryption methods for data transmission and storage. Ensuring secure communication between the Arduino and the cloud platform is crucial for protecting sensitive information.
4. Broader Application Scope:
   - Extend the system's application to other types of electrical measurements, such as power quality analysis and energy consumption monitoring. Adapting the system for different sensor types and measurement ranges can broaden its utility in various IoT scenarios.
5. User Interface Improvements:
   - Develop a more intuitive user interface for configuring and monitoring the system. This could include mobile applications or web dashboards providing real-time feedback and control over the measurement process.

In conclusion, this project demonstrates the feasibility and effectiveness of using Arduino Mega and NB-IoT technology for real-time voltage and current measurement in an AC network. The integration with ThingSpeak provides a robust platform for data storage, visualization, and analysis. By addressing the identified challenges and exploring the recommended future directions, this system can be further enhanced to meet the growing demands of IoT applications in electrical parameter monitoring.

REFERENCES

Artificial intelligence has been used in the work as follows:
ChatGPT 2024. OpenAI. GPT-4o. Assessed for Language check, May 2024.
https://chat.openai.com
The above information is formed from the following details:
Service year. Company. Version. How utilized, when. Web address.

Adhikary, A. L. 2016. Performance evaluation of NB-IoT coverage. In 2016 IEEE 84th Vehicular Technology Conference, 1-5.

Alahakoon, D. &. 2015. Smart electricity meter data intelligence for future energy systems: A survey. IEEE transactions on industrial informatics., 425-436.

Al-Fuqaha, A. G. 2015. Internet of Things: A survey on enabling technologies, protocols, and applications. IEEE communications surveys & tutorials, 2347-2376.

Analog. 2024. Online publication. https://www.analog.com/en/products/ds3231.html. Accessed 22.04.2024

Arduino. 2022. Online publication. https://docs.arduino.cc/learn/built-in-libraries/software-serial/. Accessed 22.04.2024

Arduino. 2024. Online publication. https://www.arduino.cc/reference/en/language/functions/time/millis/. Accessed 22.04.2024

Atzori, L. I. 2010. The internet of things: A survey. Computer networks, 2787-2805.

Azure, M. 2016. Online publication. https://docs. microsoft. com/eses/azure/virtual-machines/linux/quick-createportal. Accessed 19.05.2024

Badamasi, Y. A. 2014. The working principle of an Arduino. In 2014 11th international conference on electronics, computer and computation, 1-4.

Chen, M. M. 2017. Narrow band internet of things. IEEE access, 20557-20577.

ChatGPT. 2024. AI Tool. https://chatgpt.com/. Accessed 01.02.2024

Da Xu, L. H. 2014. Internet of things in industries: A survey. IEEE Transactions on industrial informatics., 2233-2243.

Debele, G. M. 2020. Automatic room temperature control system using Arduino Uno R3 and DHT11 sensor. In 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing, 428-432.

Devalal, S. &. 2018. LoRa technology-an overview. In 2018 second international conference on electronics, communication and aerospace technology, 284-290.

DFRobot. 2018. Online publication. https://github.com/DFRobot/DFRobot_SIM7000. Accessed 19.05.2024

Dfrobot. 2018. Online publication. https://wiki.dfrobot.com/SIM7000_Arduino_NB-IoT_LTE_GPRS_Expansion_Shield_SKU__DFR0505_DFR0572. Accessed 19.05.2024

Fluke. 2024. Online publication. https://www.fluke.com/en-us/product/accessories/current-clamps/fluke-i30s#. Accessed 19.05.2024

Fransiska, R. W. 2013. Electrical power measurement using arduino uno microcontroller and labview. In 2013 3rd international conference on instrumentation, communications, information technology and biomedical engineering, 226-229.

Gubbi, J. B. 2013. Internet of Things: A vision, architectural elements, and future directions. Future generation computer systems, 1645-1660.

Higham, D. J. 2016. MATLAB guide. Society for Industrial and Applied Mathematics.

Horowitz, P. H. 1989. The art of electronics . Cambridge: Cambridge university press.

instruments, S. 2024. Online publication. https://www.sapphire.com.tw/products_detail/3.htm Accessed 19.05.2024

Kandimalla, J. &. 2017. Web based monitoring of solar power plant using open source IOT platform Thingspeak and Arduino. International Journal for Modern Trends in Science and Technology, 3-4.

Kelechi, A. H. 2022. Design of a low-cost air quality monitoring system using Arduino and thingspeak. Comput. Mater, 151-169.

Kusriyanto, M. &. 2016. Smart home using local area network based arduino mega 2560. In 2016 2nd international conference on wireless and telematics, 127-131.

Lauridsen, M. K. 2016. Coverage and capacity analysis of LTE-M and NB-IoT in a rural area. In 2016 IEEE 84th Vehicular Technology Conference, 1-5.

Litwin, L. 2000. FIR and IIR digital filters. IEEE potentials, 28-31.

Ma, Y. Z. 2019. WiFi sensing with channel state information: A survey. ACM Computing Surveys, 1-36.

Malik, P. K. 2022. Narrow band-IoT and long-range technology of IoT smart communication: Designs and challenges. Computers & Industrial Engineering, 172, 108572.

Manyika, J. C. 2015. The Internet of Things: Mapping the value beyond the hype. McKinsey Global Institute.

mathworks. 2024. Online publication. https://se.mathworks.com/help/thingspeak/. Accessed 19.05.2024

Maureira, M. A. 2011. ThingSpeak–an API and Web Service for the Internet of Things. World Wide Web, 1-4.

Mehmood, Y. A. 2017. Internet-of-things-based smart cities: Recent advances and challenges. IEEE Communications Magazine., 16-24.

Morello, R. D. 2017. The role of advanced sensing and IoT in the electric grid of the future. IEEE Sensors Journal, 7828-7837.

OpenAI. 2024. Online publication. https://openai.com. Accessed 01.02.2024

Palattella, M. R. 2016. Internet of things in the 5G era: Enablers, architecture, and business models. IEEE journal on selected areas in communications, 510-527.

Pasha, S. 2016. ThingSpeak based sensing and monitoring system for IoT with Matlab Analysis. International Journal of New Technology and Research, 19-23.

Pelgrom, M. J. 2013. Analog-to-digital conversion. New York: Springer New York.

Por, E. v. 2019. Nyquist–Shannon sampling theorem. Leiden University, 5.

Pulver, T. 2019. Hands-On Internet of Things with MQTT: Build connected IoT devices with Arduino and MQ Telemetry Transport. Packt Publishing Ltd.

Raza, U. K. 2017. Low power wide area networks: An overview. ieee communications surveys & tutorials, 855-873.

Ripka, P. &. 2010. Advances in magnetic field sensors. IEEE Sensors Journal, 1108-1116.

Sigrist, L. G. 2017. Measurement and validation of energy harvesting IoT devices. In Design, Automation & Test in Europe Conference & Exhibition, 1159-1164.

Sinha, R. S. 2017. A survey on LPWA technology: LoRa and NB-IoT. Ict Express, 14-21.

Söderby, K. 2024. Online publication. https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-serial-monitor/. Accessed 19.05.2024

Toutsop, O. K. 2021. A comparative analyses of current IoT middleware platforms. In 2021 8th International Conference on Future Internet of Things and Cloud, 413-420.

Tutorialspoint. 2024. Online publication. https://www.tutorialspoint.com/structs-in-arduino-program. Accessed 19.05.2024

Walden, R. H. 1999. Analog-to-digital converter survey and analysis. IEEE Journal on selected areas in communications, 539-550.

Yin, J. M. 2022. Application Scenarios and Analysis of NB-IoT Communication in Substation and Power Internet of Things. In 2022 IEEE 5th International Electrical and Energy Conference, 838-842.

Zafar, S. M. 2018. An IoT based real-time environmental monitoring system using Arduino and cloud service. Engineering, Technology & Applied Science Research, 3238-3242.