CONTINUOUS REAL-TIME WIRELESS MONITORING OF SUBSTATIONS USING CELLULAR COMMUNICATION

An Undergraduate Research Scholars Thesis

by

KHALID DARWISH, JUDE AHMED, TAMIM AL TAMIMI, AND MAYAR MAHMOUD

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Dr. Ali Ghrayeb

May 2024

Majors: Electrical Engineering

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

We, Khalid Darwish, Jude Ahmed, Tamim Al Tamimi, and Mayar Mahmoud, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with our Faculty Research Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

ABSTRACT

Continuous Real-Time Wireless Monitoring of Substations Using Cellular Communication

Khalid Darwish, Jude Ahmed, Tamim Al Tamimi, and Mayar Mahmoud
Department of Electrical and Computer Engineering
Texas A&M University

Faculty Research Advisor: Dr. Ali Ghrayeb
Department of Electrical and Computer Engineering
Texas A&M University

The main objective of this research thesis is to provide real-time monitoring of the status of substations of power systems by wirelessly transmission from a substation to a control center in real time. This will not only reduce the amount of labor that is being deployed at each substation to record data manually but also replace wired communication with wireless using Long-Term Evolution (LTE)/4G Technology. The process starts from the substation, the data and readings are sent to the microcontroller that works on CANbus and Modbus interfaces which then sends the data being collected in real-time to the server. The server processes and stores the data in the database, waiting for a request from the client to respond with the needed data. After the request is sent from the client, who in this case is the web interface, to the server, then a response is sent to the web interface. The data is received by the web interface and is further processed and displayed on the browser as plots or tables. This allows engineers to virtually monitor the status of substations easily without having to physically visit the substations to get the information needed. Multiple steps were taken with the integration of the data collected from the microcontroller to the server, and corresponding ports were used for the web app and the

microcontroller to allow them to communicate in real-time. Encryption will be used to make sure all the data that is traveling between the components is protected and no other external entity can access them.

ACKNOWLEDGEMENTS

Contributors

We would like to thank our faculty advisor, Dr. Ali Ghrayeb for his guidance and support throughout the course of this research.

Thanks also go to our friends and colleagues and the department faculty and staff for making our time at Texas A&M University a great experience.

Finally, thanks to our mentor Naheel Kamal for his encouragement and advice in this project.

All other work conducted for the thesis was completed by the team independently.

Funding Sources

No funding was received for this project.

1. INTRODUCTION

The Continuous Wireless Monitoring of Substations Using Long-Term Evolution (LTE)/4G Technology enables the status of substations to be wirelessly transmitted from the substation to a control center in real time. Each substation can be monitored remotely in real-time with this design, which includes a microprocessor with a built-in modem that sends signals to an application. This application would allow the user, a substation engineer, to monitor and control the system from any location in real-time. This communication is carried out via a server, which allows communication between the application and the microprocessor, as well as a strong and efficient encryption algorithm to ensure system security. This monitoring system will monitor multiple data sets at the substation to help avoid any malfunctions within the substation that may result in power outages in cities. Cryptography will be implemented onto the data being transmitted to ensure no outside entities can access or interfere. This cryptography will include a hybrid implementation of both symmetric and asymmetric encryption types to provide optimal security without time delay.

Previously General Packet Radio Service (GPRS) technology, a packet-switching technology that enables information to be transmitted through mobile networks, has been used for substation monitoring. However, GPRS offers very low data rates and there is limited literature on the security aspect of the wireless communication. GPRS devices can communicate using IP address mapping technology via an internet server, allowing data packets to be sent quickly and easily across existing networks [1]. GPRS devices have a variety of capabilities, including communication with one another, short message transmission, and communication mode integration. Specifically, GPRS technology has been used to monitor substations using

mobile monitors and short message transmission. Every substation would be equipped with monitoring equipment. This apparatus monitored the temperatures of transformer oil, temperature, and humidity. The status is then periodically sent to the control center. The control center can then issue commands to the GPRS devices via short message transmission. The hardware used in previous literature consists of three major components: the measuring, the microprocessor, and the GPRS module. The measuring part measures current, voltage, temperature, and humidity [2]. The microprocessor module also contains the algorithm that handles data acquisition, processing, display, transmission, and reception. Finally, the GPRS device is integrated to transmit messages. Although this technology allows for data transmission wirelessly, it lacks the security and speed that could be achieved with newer technology. Not only that, but these implementations of wireless sensor networks have large security threats [3]. With stronger security systems comes a time delay, and in the scenario of real time data monitoring, a limited time delay is imperative. Hence, this project aims to determine a strong encryption system for data being communicated wirelessly without causing significant time delays.

2. METHODS

2.1 Hardware Components

The major hardware components used for the project consists of 5 parts, the substation equipment, the microcontroller, an electromechanical relay, a MAX485 module, and an MCP2515 module. The readings of voltage, current and power being distributed to the neighboring buildings would come from substation equipment, the readings would be processed through either the MAX485 module, or the MCP2515 module, this depends on the interface that the substation's equipment, operates on. The MAX485 module allows the microcontroller to interface with equipment that utilize Modbus, and the MCP2515 does the same but only with the CANBus interface, without these modules, the microcontroller would not be able to receive the voltage, current, and power readings. Once the microcontroller receives the data, the ESP32 module on the microcontroller would process, and encrypt the data using an AES encryption algorithm. The LTE module on the microcontroller would send the encrypted data to the backend server via WebSocket. The microcontroller will use the LTE module to connect to a public server that would be accessible to the engineers that are monitoring the substation. Finally, the electromechanical relay is used as an option to turn the microcontroller on and off, the command to operate the relay would come from an administrator using the WebApp. Figure 1 shows a general configuration that the hardware components would be connected together for the final operation of the monitoring system.

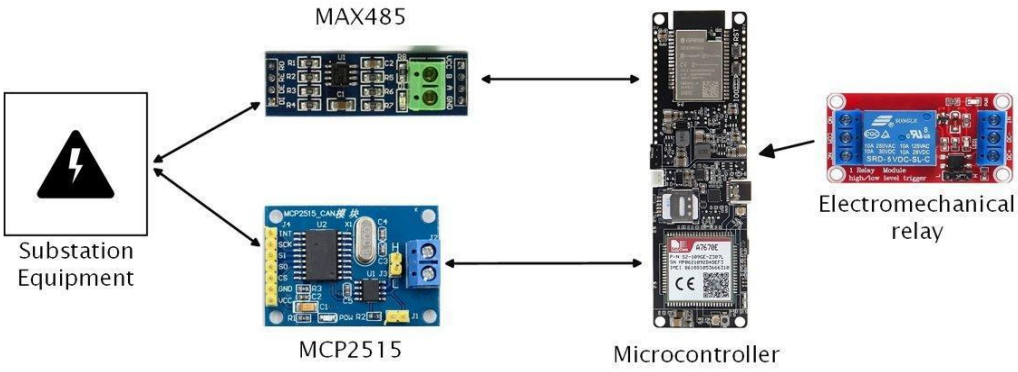

Figure 1: The configuration of the hardware components for the final system.

2.2 Back-end Server

Receiving data from the microcontroller and sending it to the web application is the backend server's primary job. For more explanation, data is formatted by the microcontroller upon receiving it from the substation and is subsequently delivered to the server through the WebSocket protocol. After receiving and processing the data, the server saves it in a database. The web application may use HTTPS and WebSocket to retrieve historical and real-time data from the database saved on the server. The client can access data using this web application, which employs real-time communication between it and the client [4].

Hypertext Transfer Protocol Secure (HTTP) is used by the web interface, while Espressif Systems 32 (ESP32) is used by the hardware, making up the server's two clients. Since Node.js is the most popular framework for developing servers with JavaScript, it will be specifically utilized to create the server in that programming language [5]. Based on the research done on various libraries, Express.js and WebSocket are the libraries that implement the communication protocols. Express.js is a web application framework for Node.js to construct server-side websites and application programming interfaces. Node.js can create server-side webpages and application programming interfaces using the Express.js web application framework. Express.js

follows the request-response communication method, in which a client sends an HTTP request, and the server replies with a corresponding answer; this protocol handles the requests independently. Furthermore, a client and server can communicate in bidirectional real-time via the WebSocket protocol, eliminating the need to wait for a response or request [6]. Web Socket is best suited for applications like communication apps that require instantaneous data updates, and it operates using the WebSocket protocol. Lastly, WebSocket will be utilized for communicating with the microcontroller and Express.js will be utilized for communicating with the web interface.

Fig. 2 below illustrates the communication between the server and the client. The server, as shown, includes one HTTP and two Web Sockets, each of which communicates with a different media. The port numbers assigned are 8000, 5050, and 8080, respectively. The HTML file, or web page document, has a JavaScript file inside of it that contains the code that operates on the client, which in this case is the browser. The web application and the server communicate in this way: the HTTP sends an HTML file to the client which has the JavaScript file, and then the JS file communicates with WebSocket 1. The server communicates with the client through WebSocket 2.

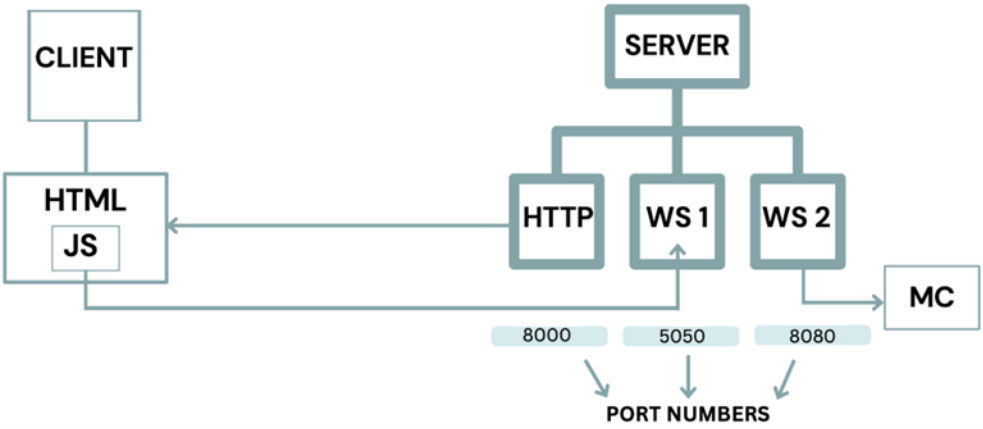

Figure 2: Breakdown of server-to-client communication between microcontroller and web interface.

As of now, while establishing the connection between the microcontroller, server, and web interface, the IP address of the server is used by the microcontroller and web app. However, a public virtual private server will be used in order for the web app to be accessible to the engineers wherever they are. A virtual private server, or VPS, operates by dividing a physical server into several separate virtual environments, each of which has its own resources, operating system, and customizations. With the performance, management, and isolation of a dedicated server without the high costs and complexity of managing actual hardware, a virtual private server (VPS) offers an affordable and adaptable hosting option. It is commonly used for hosting websites, apps, databases, and other internet services by people, companies, and organizations. In addition to that, a reverse proxy will be used to improve the overall system in terms of performance, security, and reliability. When used in such a way, Nginx serves as a middleman server between clients and backend servers, a configuration known as a reverse proxy. Clients connect to Nginx instead of directly to backend servers, and Nginx uses established rules to redirect requests to the suitable backend server. A client, for instance, a web browser, connects to the reverse proxy server (Nginx) by utilizing its address, or domain name, when it submits a request to access a resource. Nginx decides which backend server is responsible for handling the request based on established rules such as URL routes, domain names etc. After being analysed, Nginx receives the response from the backend server and sends it back to the client that made the initial request. To put it in simple terms, Nginx effectively controls the flow of communication, guaranteeing smooth communication between the network's clients and backend servers.

2.3 Web Application

The web application aims to provide the user with full access to monitor the substation equipment with full security and minimal time delay. Plots are also provided in real-time

monitoring, which allows the user to continuously monitor many subcomponents of the substation. In specific, the web application starts with a login page, and the credentials provided are verified with the data that is stored in the database backend server. There will be multiple monitoring options that include voltmeter readings, ammeter readings, and circuit breaker readings. There will also be an option for the overall control of the substation, and that means that the user can switch the substation on or off, in the case that the monitoring shows a fault and requires that the substation needs to be shut down. Having a circuit breaker reading option is extremely crucial since it is an electromechanical device that protects electrical devices that are used in: main distribution boards which are widely available in most of the houses, feeder pillars. Specifically, it functions by blocking the electric current that exceeds the specified standards, and it also limits the supply of energy that is being delivered to the load. A voltmeter is an essential option since it is used to measure the electric potential difference between two different points, and is also used to measure sinusoidal AC and DC voltages. Also, the use of an ammeter option is crucial since it works by using the magnetic field around a specified coil, and usually the strength of the magnetic field varies differently with the magnitude of the current. Finally, an option for the substation's history readings for the subcomponents will be available.

The main objective for creating a web application is that it can be easily accessed by multiple devices simultaneously, and it is chosen over a mobile application since it is easier to read the plots and the information that is provided by the substation’s subcomponents. It also has better discovery options on search engines which makes it essential since the plots will be displayed on a public server, and another reason for going with the web application option is for its cost effectiveness due to its smaller amount of development time required. Moreover, many frameworks can be used for this process, but the most suitable framework is Express.js since it

can exchange the data as efficiently as possible between the backend server and the web app over Web Sockets and HTTP. Express.js is also unopinionated which means that it has no limitations over the structure that is used, and the middleware is used to verify the requests and responses that are usually being exchanged between the web app and the backend server. The usage of Express.js is also good for its forms of minimality, flexibiltiy, and its performance. From the minimality perspective, it is good since it meets the client's needs with the required functionalities. Additionally, the data is being exchanged over Web Sockets since it does not require request or response procedures, and that is essential to reduce the buffering times that the user can experience. The procedure usually starts with the data being sent from the microcontroller to the backend server, and then to the web app in which data can be read from the plots. Finally, all of this will be done on the same IP address since the web application plays a vital role in requesting all of the data from the backend server.

Considering the importance of the web application, a home page was constructed with multiple options that represent the substations subcomponents, and it shows the plot that is being displayed in real time monitoring. As it was mentioned previously, real time monitoring is a crucial part of receiving the data from the backend server. The data is being sent at a rate of 1ms since there should be a minimal time delay connection between the backend server and the web application. Furthermore, Plotly library was used in JavaScript with its corresponding dependencies in which it can be the best option for large data sets since it allows hovering to detect any outliers or anomalies. Plotly is an open source library which means that it can be used for data visualization, and allows the user to have multiple options for the customization of the plot through the plot itself or by modifying the plots axes on the HTML code [7].

An HTML code was built to display the data originally transmitted from the substation to the backend server and then the web application in real time. The HTML code is being built from the meta tags that include the characters encoding and viewports settings. Moreover, it creates a websocket connection between the backend server and the web application on port 127.0.0.1:5052 accordingly with arrays of time stamps and values that peak at 100V. 'update chart' is used to update the chart every 1ms with that data that is being received from the backend server that re-renders the plot. An 'event listener' was set to parse the data received as Java Script Object Notation (JSON) which is a standardized text format that displays structured data in a readable format for the user [8]. Furthermore, the next step is to configure and enhance further on the codes to obtain a sinusoidal wave with a bandpass filter, low pass filter, and a high pass filter to eliminate any noise disturbances. Having a smooth sinusoidal is extremely important since it can have better flow for the client to be able to analyze and read, and it is also important to link the login page, home page all together with the plot to make it as accessible as possible for the client to configure on. Finally, all the options that the client will go over should be fully connected in real time with the substation no matter how far the distance is since long term evaluation (LTE) is being established to clear out the task of distance. The figures below show the HTML code of the client, and the home page that displays the multiple options that the client can choose from. There will also be more options to provide multiple users to have different access levels since it is important to split the tasks that can be accessed by different users at different times, and that means that a user can have full direct operation of the substation while another user has limited access to only the monitoring features of the substation. There will also be alert notifications that are set up for critical events that can include circuit breaker trips. Furthermore, there will be a performance monitoring option that tracks the subcomponents

response times, server loads, and most importantly the network latency. Finally, the web application is ready with all of these options to ensure the user has smooth control over the plots to be able to receive any issues that can occur, and have the ease to troubleshoot based on the cruciality of the issues.

```
<script>
    const socket = new WebSocket('ws://localhost:5052');

    // Initialize empty arrays for x and y data
    let timestamps = [];
    let values = [];

    // Create a new Plotly chart
    const layout = {
        title: 'Real-Time Data Visualization',
        xaxis: {
            title: 'Time'
        },
        yaxis: {
            title: 'Value'
        }
    };

    Plotly.newPlot('myChart', [], layout);

    // Function to update the chart with new data
    function updateChart(timestamp, value) {
        timestamps.push(timestamp);
        values.push(value);

        Plotly.newPlot('myChart', [{
            x: timestamps,
            y: values,
            type: 'scatter',
            mode: 'lines+markers'
        }], layout);
    }

    // Event listener for WebSocket messages
    socket.addEventListener('message', (event) => {
        const data = JSON.parse(event.data);
        updateChart(data.timestamp, data.value);
    });
</script>
</body>
</html>
```

Figure 3: HTML code of the client's side.

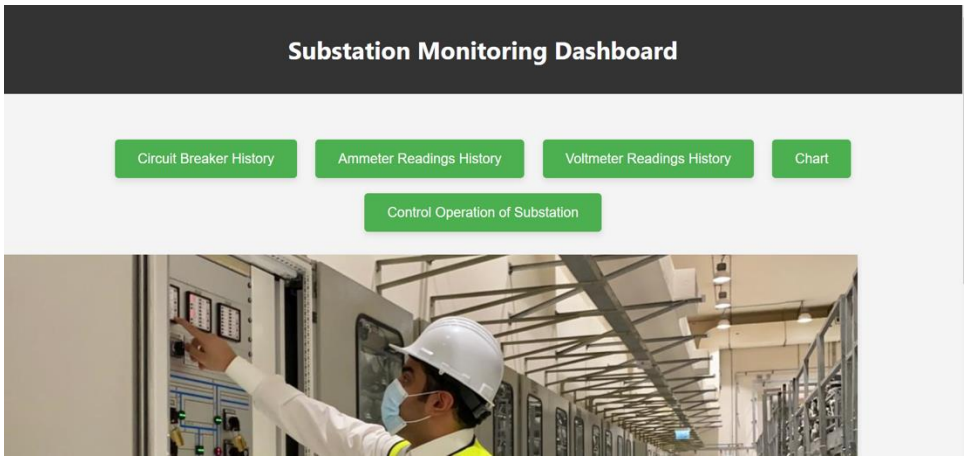

Figure 4: Home Page with multiple substation components options.

2.4 Encryption

Because data will be exchanged bidirectionally between the microcontroller, the server, and the interface, it is imperative that cryptography is applied onto the data so that outside sources cannot access or interfere with the data being sent. A security layer will be applied to the

communicated data utilizing both an Advanced Encryption Standard (AES) symmetric algorithm and Elliptic Curve Cryptography (ECC) for the asymmetric part, specifically the key generation. Because the server and app interface software will both be implemented on JavaScript (i.e., Node.js), one implementation of the encryption algorithm will be written in JavaScript. The other implementation will be written in C++ because that is the software on which the microprocessor will be based. These two encryption algorithm implementations will require secure encryption keys, which will be used for both encryption and decryption. The same key will be used at both ends of the process, and the key will be generated using ECC. For the ECC key generation, the Diffie Hellman process is currently being explored, which involves two parties agreeing on a shared key over an insecure channel. In figure 3, the entire encryption and decryption process is displayed between Bob and Alice, respectively the sender and receiver. A random set of numbers will be used as the shared password, it is initially encrypted using Alice's public key, and the ciphertext is then sent to Alice to be decrypted with her private key. At this point, Bob, and Alice exclusively both have this password.

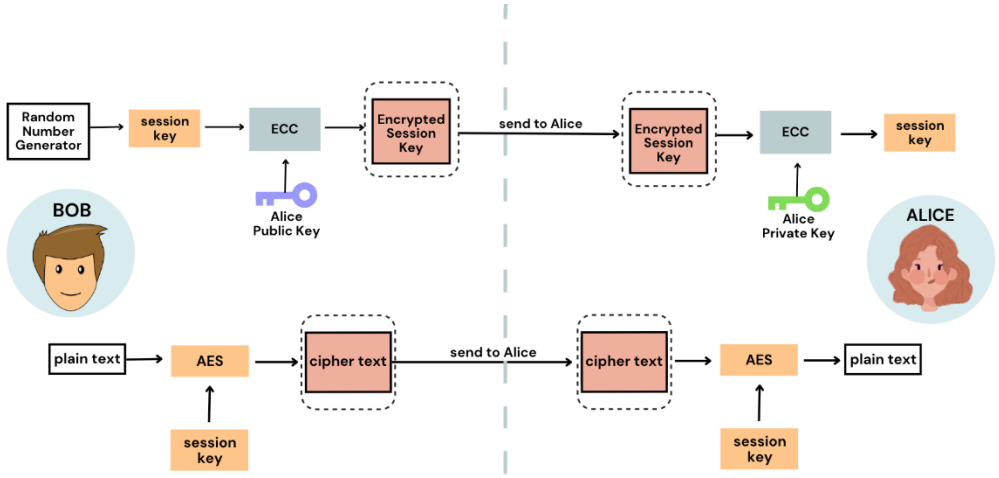

Figure 5: AES and ECC encryption implementation between Bob (sender) and Alice (receiver).

From there, the asymmetric AES encryption can take place on the data using this shared private password as shown in Figure 3. With Bob, the plaintext is encrypted using the shared password and the ciphertext can be sent to Alice and decrypted at the destination with the shared password. The public key will be generated from Alice's private key and will be stored using Public Key Infrastructure.

3. RESULTS

3.1 Microcontroller

The ESP32 microcontroller's LTE connection had a very limited range when we implemented the project due to the sim card that was used in our implementation, the range of the LTE connection was limited to a small area of a few meters which is clearly not ideal for the application that we envisioned at the start of the project, that being, presenting an alternative monitoring system to the currently implemented monitoring system that is used for an electrical substation at the distribution level. However, given that constraint, the implementation of the project has shown very promising results, as the LTE connection between the microcontroller and the server utilizes the encryption algorithms of AES and ECC, to transmits the data rapidly and securely.

For the project implementation, the microcontroller was tested using data that was created specifically for the test. The data that the microcontroller was transmitting during the implementation consisted of time in microseconds, and three phase voltage and current signals. The microcontroller was not tested with authentic data from an electrical substation, because authentic data of electrical substations was not readily available. Furthermore, the physical interfacing of the microcontroller with the substation equipment was never tested as we were not able to obtain access to the equipment of a substation.

Figure 6 shows the data that the microcontroller transmits when the whole system was implemented. The data is synthetic and represents values for time, three phase voltage and current. The printed data shown in the figure goes as follows, every row shows the time in microseconds, voltages in phases A, B and C, then the current for phases A, B, and C. The data

was generated at a constant stream with a sampling rate of 10 milliseconds. The microcontroller sends all of this data to the server, and the server receives the data in real time, all of this occurs while the encryption algorithm is running behind the scenes ensuring the security of the transmitted data between the microcontroller and the server.

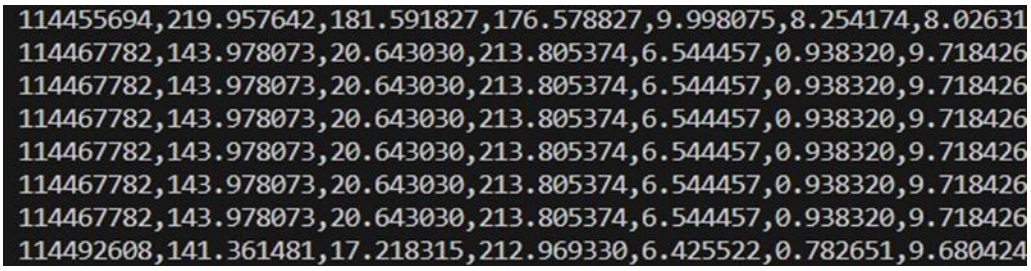

Figure 6: The data that was sent from the microcontroller to the server.

3.2 Backend Server

A code for the server was constructed to receive data from the microcontroller in real-time and store it in the database so that when the browser requests it, it responds with the data that was requested.

```
console.log("Starting...");

//event listener for connection ws8080, client connects
wss8080.on('connection', function connection(ws) { // from esp
  console.log("Connected");
  ws.on('error', console.error);
  ws8 = ws;

  //event listener for message, received, data to string, checks if ws5 is defined, sends message
  ws.on('message', function message(data) {
    data = data.toString();
    let decrypted = decrypt_uc(data, key_uc)
    if (ws5 != undefined) {
      let encrypted = encrypt_web(decrypted, key_web)
      ws5.send(encrypted);
      console.log(data);
    }

  });
```

Figure 7: Server code which sends and receives data.

The code starts by printing “Starting…” when the script is starting to execute. After that, an event listener for the connection from WebSocket 8080 is established, and when the client is successfully connected, it prints “Connected”. In order to process incoming messages, this line configures an event listener for the WebSocket connection. Upon receiving a message from the client (from the microcontroller), this callback method will be performed. The message that was sent by the client is represented by the parameter data. If the data being transmitted is binary digits, it can be converted to a string. The data received is going to be fully encrypted, therefore, it must be decrypted using the microcontroller key. When the web interface requests data, it is encrypted again using the web interface key before responding.

Currently, the data used is synthetic data to test the communication between the microcontroller, server, and web app. As shown in the figure below, the data was successfully received in real-time from the microcontroller and stored. The microcontroller and the HTML file for the web interface used the same.

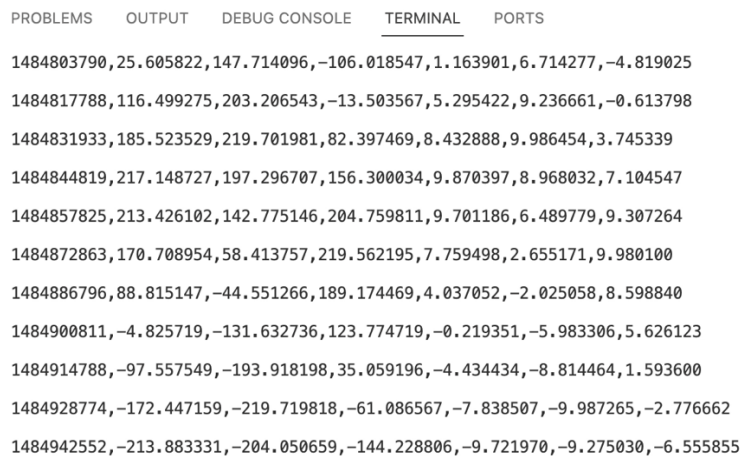

Figure 8: Successful real-time data transmitted between the microcontroller and server.

As previously stated, the server code was initially created to test and validate the connectivity between the complete system, which included the microcontroller, web application,

and encryption. When the server goes to a public platform, its scalability and reliability improve, as does the simplicity of the connecting process. Because it is hosted on a public server, the server has access to reliable resources, ensuring consistent performance even during peak traffic or demand periods. The server may now operate on a public server after the connections are successfully established. The microcontroller and web application will be able to connect to the server without needing to provide a specific IP address that matches the server, which will be beneficial for the system. Additionally, it would make it possible for any engineer, regardless of location, to virtually monitor and track the substation's status in real-time without any interruption. This approach, after all, guarantees a high degree of ease of use and accessibility for remote monitoring, giving engineers command over important tasks.

```
mayarelazab@altair Code2 % scp –i ~/.ssh/mayar –r ./* mayar@naheel.xyz:/home/mayar/Code2/
```

Figure 9: Transmitting local server to a public server.

First, the code above is implemented as it can safely transfer several folders and files from one computer to another. SCP (Secure Copy Protocol) is being used by the given command to safely move files and directories from the local computer to the far server. Initially, the SCP command (scp) is executed, with the identity file (-i ~/.ssh/mayar) containing the private key for authentication specifically specified. Recursive copying is indicated with the -r parameter, which allows whole directories to be transferred. ./.* indicates the source of the transfer, which includes all files and folders in the present directory (./). Recursive copying is indicated with the -r parameter, which allows whole directories to be transferred. *./.* indicates the source of the transfer, which includes all files and folders in the present directory (./). Lastly, it indicates that the transfer will go to mayar@naheel.xyz:/home/mayar/Code2/. With the hostname naheel.xyz

and the username mayar, this implies that the files will be copied to the remote server's /home/mayar/Code2/ directory.

```
[mayarelazab@altair ~ % ssh -i ~/.ssh/mayar mayar@naheel.xyz
 Last login: Tue Mar 26 11:47:04 2024 from 37.186.46.142
```

Figure 10: Establishing a connection with the public server.

Similarly, this code is written after the previous code to establish a connection with the public server. To begin, the SSH (Secure Shell) command securely connects to a remote server called "naheel.xyz" with the username "mayar". The next part is the private key, which is used for verification, indicated by -i ~/.ssh/mayar, where it doesn’t require a password and guarantees that only authorized users can access it by supplying this key. The last part is the server denoted by naheel.xyz and the user account denoted by mayar in which the connection was successfully established.

3.3 Encryption Algorithm on JavaScript

The first implementation of the encryption algorithm for this project was developed on JavaScript, as the communication between the server and the web applications will be conducted on the JavaScript programming language. To ensure secure communication between these two components, a hybrid of symmetric and asymmetric encryption is applied onto the data at the source and decrypted at the receiver. This process begins with the use of ECC to generate the key to be used for symmetric encryption later.

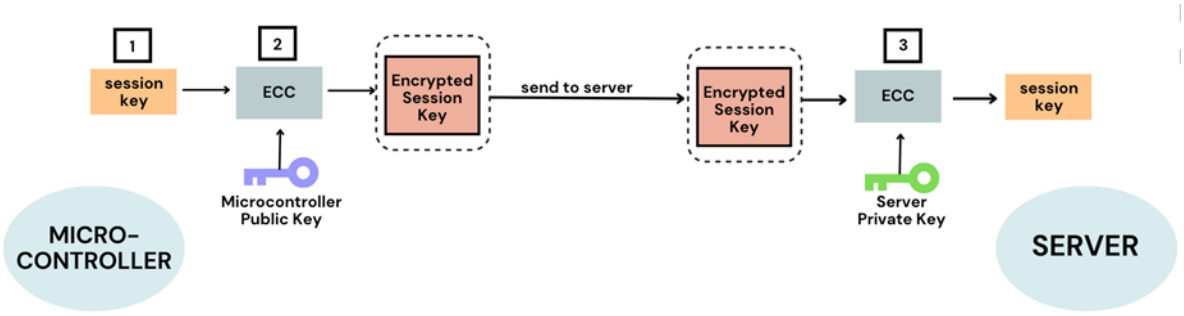

Figure 10: Overview of the asymmetric encryption process.

In step 1 of Figure 10, the session key is generated using elliptic curve 25519 which offers a 128 bit security key which is shown in the first two lines of the program in figure INSERT. Afterwards a key pair is generated, the public key is stored in Public Key Infrastructure while the private key is known only to the microcontroller.

```
var EC = require('elliptic').ec;
var ec = new EC('curve25519');
//curve 25519 offers 128 bit security (256 bit key) and is used for ecdh
// Generates key private and public key pair for key 1 and 2
var key1 = ec.genKeyPair();
var key2 = ec.genKeyPair();
```

Figure 11: Program for the ECC key generation.

At this point, the third step of the asymmetric encryption process takes place, and the program derives a shared secret key (represented by shared1) by combining the microcontroller's private key (key1) with the server's public key (key 2). This program is displayed in figure 13 and takes place at step 2 of the encryption process in figure 10.

```
var shared1 = key1.derive(key2.getPublic());
```

Figure 12: Program for deriving the shared key for the sender using the sender's private key and the receiver's public key.

The third and final step of the key generation takes place and allows for the server to derive the same shared key. The server's private key (key1) is combined with the microcontroller's public key (key 2) to get the shared key (shared2).

```
var shared2 = key2.derive(key1.getPublic());
```

Figure 13: Program for deriving the shared key for the receiver using the receiver's private key and the sender's public key.

This process allows for key generation to take place, where a shared key (shared1 and shared2 in the program) are generated at the microcontroller and server side respectively and are then able to use that shared key to perform symmetric encryption. The symmetric encryption begins with the use of a function that calls for a key and performs encryption and decryption with that singular key.

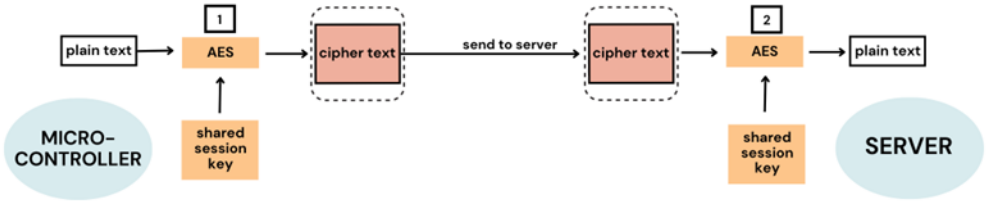

Figure 14: Overview of symmetric encryption protocol.

In Step 1 of Figure 14, the symmetric encryption takes place where the variable "key" stores the shared key that was created using the ECC program. The function encrypts "data" which is plaintext through AES encryption. This function returns an encrypted string of ciphertext which can then be decrypted in the next steps.

```
function encrypt(data, key) {
    var encrypted = CryptoJS.AES.encrypt(data, key);
    return encrypted.toString();
}
```

Figure 15: Program for the symmetric encryption function.

The output of the hybrid system (both the symmetric and asymmetric combined algorithm) shown in figure 16 portrays the success of the encryption. The shared key for both the microcontroller and the server are the same, meaning the ECC key generation was successful. Both the sender and the receiver have generated a key that is only known to each other and cannot be accessed by any outside sources. The original message that is encrypted is simply a string for experimentation purposes, and the encrypted message is also shown. The encrypted message is the ciphertext which is not legible to outside sources, and the decrypted message at the destination is the same as the original plaintext, therefore the encryption process was successful.

```
shared key for microcontroller: 339d47fbd86dd00e4d4e5c6f449047ede1d1b5e5fd04cbf000d5ed0e2606ff3a
shared key for server:  339d47fbd86dd00e4d4e5c6f449047ede1d1b5e5fd04cbf000d5ed0e2606ff3a
original message:  substation status
encrypted message:  U2FsdGVkX1/+41E+6gvI8E3qvC6ucdi57coe1JG+BjRZeOmz30s1ddrlxB8YTORy
decrypted message:  substation status
```

Figure 16: Output of the hybrid encryption program.

3.4 Encryption Algorithm on C

The C code implementation for the communication between the microcontroller and server is in its beginning stages. Currently the program displayed below performs AES encryption on a given plaintext, then decrypts it. In Figure INSERT, a test is run to check if the original plaintext is the same as the plaintext after decryption. If they are the same, then the

program prints succeeded, if they are not then it prints failed. Figure INSERT shows the successful output of the AES encryption on C program.

```
for(idx = 0; idx < 2; idx++) {
    aes_encrypt(plaintext[idx], enc_buf, key_schedule, 256);
    //printf("\nPlaintext    : ");
    //print_hex(plaintext[idx], 16);
    //printf("\n-encrypted to: ");
    //print_hex(enc_buf, 16);
    pass = pass && !memcmp(enc_buf, ciphertext[idx], 16);

    aes_decrypt(ciphertext[idx], enc_buf, key_schedule, 256);
    //printf("\nCiphertext   : ");
    //print_hex(ciphertext[idx], 16);
    //printf("\n-decrypted to: ");
    //print_hex(enc_buf, 16);
    pass = pass && !memcmp(enc_buf, plaintext[idx], 16);

    //printf("\n\n");
}
```

Figure 17: Program for the symmetric encryption on C code.

```
int aes_test()
{
    int pass = 1;

    pass = pass && aes_ecb_test();
    pass = pass && aes_cbc_test();
    pass = pass && aes_ctr_test();
    pass = pass && aes_ccm_test();

    return(pass);
}
```

Figure 18: Program to compare the original text with the decrypted text to determine if they are equal or not, this tests if the AES encryption was successful.

```
[Judes-MacBook-Pro:untitled folder judeahmed$ ./a.out
AES Tests: SUCCEEDED
```

Figure 19: The output of the AES test on C code showing the successful encryption.

The next steps for the encryption required that key generation takes place for both the sender and the receiver on C code. This key generation is imperative for both parties to have a shared key and implements a specific type of asymmetric cryptography called ECDH, or Elliptic Curve Diffie-Hellman, a specific type of elliptic curve cryptography. This key exchange protocol is based on the Diffie-Hellman algorithm and allows two parties to establish a shared secret via an unsecured channel. It is especially popular in scenarios requiring secure communication between parties, such as cryptographic protocols like TLS (Transport Layer Security). Specifically, this was implemented through three processes. The first process being a random number generator. This random number generator allows for the generation of 32 random bits to be used as the shared key between the two parties, using a seed of the current time. Then, a function called “ecdh_demo()” is used to initialize the private and public key variables for key exchange.

```
static void ecdh_demo(void)
{
  static uint8_t puba[ECC_PUB_KEY_SIZE];
  static uint8_t prva[ECC_PRV_KEY_SIZE];
  static uint8_t seca[ECC_PUB_KEY_SIZE];
  static uint8_t pubb[ECC_PUB_KEY_SIZE];
  static uint8_t prvb[ECC_PRV_KEY_SIZE];
  static uint8_t secb[ECC_PUB_KEY_SIZE];
  uint32_t i;

  /* 0. Initialize and seed random number generator */
  srand(time(NULL));

  /* 1. Alice picks a (secret) random natural number 'a', calculates P = a * g and sends P to Bob. */
  for (i = 0; i < ECC_PRV_KEY_SIZE; ++i)
  {
    prva[i] = rand();
  }
  assert(ecdh_generate_keys(puba, prva));

  /* 2. Bob picks a (secret) random natural number 'b', calculates Q = b * g and sends Q to Alice. */
  for (i = 0; i < ECC_PRV_KEY_SIZE; ++i)
  {
    prvb[i] = rand();
  }
  assert(ecdh_generate_keys(pubb, prvb));

  /* 3. Alice calculates S = a * Q = a * (b * g). */
  assert(ecdh_shared_secret(prva, pubb, seca));

  /* 4. Bob calculates T = b * P = b * (a * g). */
  assert(ecdh_shared_secret(prvb, puba, secb));
```

Figure 20: C program that creates the ECDH function and calculates the shared keys for both Alice and Bob.

Both Alice and Bob (sender and receiver) are assigned a public and private key pair and then the shared secret is calculated by Alice using Alice's private key and Bob's public key. The shared key is also generated for Bob using Bob's private key and Alice's public key. The program then checks whether the two shared keys are equal, if they are equal a statement to show that is printed as shown below in Figures 20 and 21.

```
/* 5. Assert equality, i.e. check that both parties calculated the same value. */
for (i = 0; i < ECC_PUB_KEY_SIZE; ++i)
{
  assert(seca[i] == secb[i]);
}
printf("Shared key for Alice = shared key for bob\n");
```

Figure 21: C program that tests the equality of seca (shared key 1) with secb (shared key 2) to determine the success of the asymmetric encryption.

Figure 22: Output of the C program showing the shared keys are equivalent, demonstrating the success of the asymmetric encryption.

The main function then demonstrates the ECDH key exchange and returns 0 when successful. This program displays successful key exchange between two parties, so that the shared key can then be used between the two for symmetric AES encryption. This code in combination with the previous C code for symmetric encryption develop a full cryptographic system that allow for a key to be generated and shared secretly between the sender and the receiver, and then symmetric encryption take place using that secret key when sending information between them. This can ensure that the data being sent between the microcontroller and the server is highly secure and untouchable by any outside entity.

3.5 Web Application

The client code starts with <html> as the main opening element of the whole code, and the script tag imports the plotly library as a source code to get it as an output. Furthermore, a websocket connection is established between the server and the web application to ensure it is on the same IP address, which is 127.20.10.4:5050. Moreover, there are the event listeners, which are: ‘socket.onopen’ , ‘socket.onmessage’ , ‘socket.onclose’ , ‘socket.onerror’. Finally, the ‘updateplot’ function is utilized whenever new data is received from the backend server. Figure 23 shows the real time monitored plot that was created throughout the whole process, from the microcontroller to the web application.

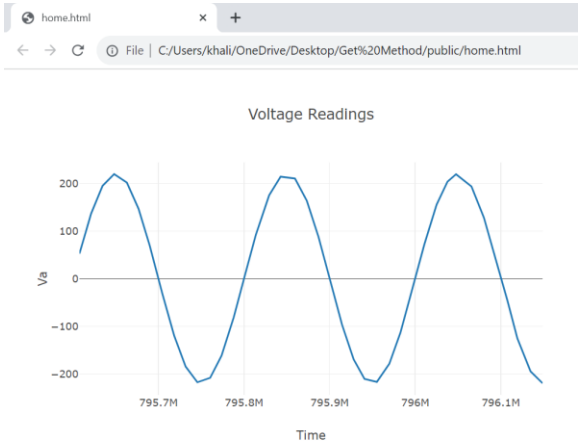

Figure 23: Real time plot that results from the explained JavaScript code above.

4. CONCLUSIONS AND FUTURE WORK

4.1 Conclusions

This work proposes a system for wireless monitoring of a substation that utilizes an LTE network connection and the WebSocket protocol for bidirectional communication between a substation and a webapp. The system consists of four main sections, the hardware, the backend server, the webapp, and finally the encryption algorithm. A microcontroller would be used to interface with the equipment inside a substation and will use LTE to send the data that it collects regarding the condition and the functionality of the substation to a backend server via a WebSocket, the backend server would receive the data from the microcontroller and store it so whenever the webapp is used, the webapp would receive the data in a visual form for the engineers to see and analyze. Throughout the entire process, an encryption algorithm that consists of a hybrid between the AES and ECC encryption algorithms would be implemented to protect the data from any cyber-attack. AES would be used for the encryption of the data between the microcontroller and the server, and between the server and the webapp while ECC would be used for the generation of the encryption keys and the sharing of the keys between the microcontroller, the server, and the webapp. The system has bidirectional communication between the sections to allow the engineers to act without the need for the engineer to be located at the substation itself.

4.2 Future Work

We propose in the future the implementation of a deep learning model into our monitoring system, the deep learning model can be created by utilizing historical data regarding the functionality of substations located within an area or country over a long period of time, the

data would need to be obtained from a local power company that would be willing to provide it. The data would be used to train the model so that the model is able to monitor the condition of the substation while also predicting and diagnosing any potential faults that may occur. The model can also provide a more efficient and effective method for monitoring a substation and a safe environment for employees working at the substation.

A deep learning model is a class of machine learning, where a computer algorithm is made to utilize data being given to it to learn, and later applies what it learned by making predictions and decisions for a given problem [9], [10]. A deep learning model can be implemented in various tasks as it can be used to identify images and videos, but it can also be used for classification of multiple data types, such as text, images, video, or sound with very high accuracies [11]. The capabilities of deep learning are always improving thanks to the constant utilization of the algorithms in various fields.

Deep learning algorithms have been utilized in multiple projects related to substation fault detection, in research done by [12], the authors applied a method for the detection and recognition of high voltage distribution frames, the goal of the algorithm was to learn how to perform task such as identifying where the key to the distribution frame would be placed, while also determining the current state of the distribution frame. Moreover, in the research developed by [13], the authors proposed a deep learning model that would be used to monitor wireless noise as well as the diagnosis of faults for a digital substation.

Based on the research that was previously mentioned it is already established that the use of deep learning algorithms to monitor substations, and for fault detection and diagnosis is very beneficial to have in order to maintain and improve upon the safety, efficiency, and effectiveness of the substation. Our inability to implement a deep learning model for the currently proposed

system comes from the lack of data that is needed to train the model in the first place, the power company that we contacted was not able to provide us with the data that we needed in order to start training the model and thus the model was put on hold until the data required can be obtained.

REFERENCES

[1] A. Nasipuri, R. Cox, H. Alasti, L. Zel, B. Rodriguez-Medina, R. Mckosky, J. Graziano, Joseph, “Wireless Sensor Network for Substation Monitoring: Design and Deployment,” *Proceedings of the 6th International Conference on Embedded Networked Sensor Systems*, 2008.

[2] L. Kong, J. Jin, J. Cheng, “Introducing GPRS technology into remote monitoring system for prefabricated substations in China,” *2nd Asia Pacific Conference on Mobile Technology, Applications and Systems*, 2005.

[3] Mazur K, Wydra M, Ksiezopolski B. “Secure and Time-Aware Communication of Wireless Sensors Monitoring Overhead Transmission Lines,” 2017.

[4] S. Nuratch, "Design and implementation of microcontroller-based platform-independent Real-time WebSocket Server for monitoring and control applications," *14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Phuket, Thailand, 2017.

[5] S. Tilkov, S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," *IEEE Internet Computing*, Dec 2010.

[6] K. Ogundeyi, C. Yinka-Banjo, “Websocket in real time application,” *Nigerian Journal of Technology*, December 2019.

[7] V. Veeramsetty, M. Kumar, S. Salkuti, “Platform-Independent Web Application for Short-Term Electric Power Load Forecasting on 33/11 kV Substation Using Regression Tree,” *Computing, Electrical and Industrial Systems*, July 2022.

[8] M. Jazayeri, "Some Trends in Web Application Development," *Future of Software Engineering (FOSE '07)*, Minneapolis, USA, June 2007.

[9] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[10] J. Schmidhuber, ‘‘Deep learning in neural networks: An overview,’’ *Neural Netw., vol. 61*, Jan. 2015.

[11] S. Oliveira, P. De Faria Neto, A. Fernandino, F. Carvalho, L. Fernandes, F. G. Guimarães, "Automated Monitoring of Construction Sites of Electric Power Substations Using Deep Learning," *IEEE Access,* 2021.

[12] Z. Fu, R. Si, H. Huang, L. Chen, J. Gao, B. Shi, C. Wang, “Research on a detection and recognition algorithm for high-voltage switch cabinet based on deep learning with an improved YOLOv2 network,” *Proc. 11th Int. Conf. Intell. Comput. Technol. Autom. (ICICTA),* Sep. 2018.

[13] Q. Zhao, L. Zhang, J. Zhao, X. Qi, W. Li, "Wireless Noise Monitoring and Fault Diagnosis for Substation Based on Ubiquitous Power Internet of Things and Deep Learning," *2019 IEEE Sustainable Power and Energy Conference (iSPEC),* Beijing, China, 2019.