Low-Cost Open Source IoT-Based SCADA System for a BTS Site Using ESP32 and Arduino IoT Cloud

Cyprian N. Oton
Department of Electrical and Computer Engineering
Memorial University of Newfoundland
St. John's, NL, Canada
cnoton@mun.ca

M. Tariq Iqbal
Department of Electrical and Computer Engineering
Memorial University of Newfoundland
St. John's, NL, Canada
tariq@mun.ca

2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) | 978-1-6654-0690-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/UEMCON53757.2021.9666691

***Abstract*—A low-cost open source IoT-based SCADA system for a rural BTS site using ESP32 and Arduino IoT Cloud is presented in this work. Current, voltage, temperature, and humidity sensors are programmed to measure relevant parameters of interest, and the measured Data is processed and parsed to the Arduino IoT Cloud via a Wi-Fi network communication channel. A widget-based dashboard is created on the Arduino IoT Cloud to monitor and control the system. A mobile application is also deployed to aid remote monitoring and control as well. LEDs are used to implement a high temperature and low voltage control logic. A prototype is used to demonstrate this as an illustration of what is obtainable in a Base Transceiver Station (BTS), where the voltage must be within a specific value (48 V) and the temperature within an acceptable value too.**

Keywords—SCADA, ESP32, Arduino IoT Cloud, sensors, Internet of Things.

I. INTRODUCTION

Supervisory Control and Data Acquisition (SCADA) is a system with a primary objective of controlling and monitoring devices in the field, mostly located in very remote and obscure places. Deploying a SCADA system on any process/plant also ensures full automation of that process/plant. Various conditions and parameters of the system can be accurately measured, monitored, and controlled optimally and in real-time. The efficiency of the process/plant is greatly improved because of the real-time nature of the data collection, processing, and carrying out the control action when necessary. Since these are achieved automatically, the chance of failure is minimized as human errors which had hitherto been prevalent are eliminated.

For a mobile telecommunication site located in a very remote area, a reliable SCADA system cannot be overemphasized as a human operator cannot be stationed at the site continuously for monitoring. With an effective SCADA system, the frequency of visits by the operators is reduced significantly, thereby reducing the overall operation cost of the site.

SCADA leverages the coalescence of hardware components like sensors and actuators and software programs like the Human Machine Interface (HMI) to perform its four primary functions: data acquisition, networked data communication, data presentation, and monitoring and control [1, 2]. To carry out these functions effectively, SCADA relies on various elements. These elements are Field Instrument Devices (FIDs) like sensors and actuators, Remote Terminal Units (RTUs) like microcontrollers and microprocessors, Master Terminal Units (MTUs) in this case, the Arduino IoT Cloud and the communication network in the case a Wi-Fi network provided by my home router. This work is based on the fourth generation of SCADA architecture which is the Internet of Things (IoT) configuration. IoT SCADA have several advantages, including remote assess/control, real-time monitoring and alarming, data sharing, data manipulation and visualization, system optimization, trend analysis, flexibility, and increased productivity. In [3, 4], our previous work on designing, sizing, and dynamic modeling of a DC hybrid power system for a remote telecommunication facility in Nigeria was examined. This work presents the SCADA aspect of the work using a prototype.

II. LITERATURE REVIEW

Monitoring, control, and instrumentation is a very crucial aspect of mobile base station sites operation. Several parameters are simultaneously monitored for effective and optimal operation with limited downtime. The significant parameters of interest are the voltage and temperature, which have a direct impact on the site. The voltage must stay within a stipulated range of value, and for a temperate region like, Nigeria, the temperature must be monitored and controlled for continuous operation of the site. The SCADA system currently operational on the site is proprietary with all the associated disadvantages and costs.

Several studies have been presented using a Web-based SCADA system. In [5], the authors presented a Web-SCADA system to monitor and control a Wind-PV power system using the IntegraXor software to create and view the graphical interface and ATMEGA8535 microcontrollers interfaced with several sensors to measure relevant data. The IntegraXor software is complicated to understand. In [6], a Web-based, low-cost SCADA system was applied to control a renewable energy system microgrid. Arduino boards were used to measure relevant electrical and environmental data using appropriate sensors. The measured data was sent to a local database hosted on a Raspberry Pi. A wireless radio frequency transceiver created the communication link between the microcontroller and the Raspberry Pi. The system is very complex to build, demanding the knowledge of HTML, MySQL, and lots of C++ codes. These can be very hard to implement.

In [7], a SCADA system to monitor the electrical parameters of a stand-alone solar photovoltaic system is presented. This work focuses on the electrical parameters of the solar system and battery efficiency. No environmental parameter is considered, and there is no room for remote control of the system. The authors in [2] developed IoT based open source SCADA system for a PV system to monitor solar panel current, voltage, and backup battery voltage. It uses Arduino Uno as the remote terminal unit for receiving the measured sensor data, and a Raspberry Pi is used as the communication channel to parse the measured

978-1-6654-0690-1/21/$31.00 ©2021 IEEE

data to Emoncms platform for storage, monitoring, and control.

In all the work reviewed, there are minor limitations that this paper seeks to address. Implementing this work on the Arduino IoT Cloud platform offers a significant level of simplicity as the code is not written from scratch. Significant code is pre-written by just declaring the variables.

III. System Description and Experimental Design

The simplified version of the studied DC hybrid system for a rural BTS site is illustrated in Fig. 1. Some parameters of interest for monitoring purposes are highlighted, albeit not exhaustive of what is typically monitored and controlled on the site. The connected components have local controllers that can receive a control signal remotely and carry out the control action via an actuator. The power management system implements the supervisory control.

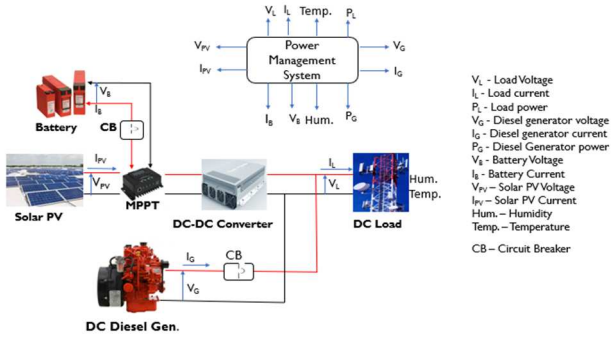

Fig. 1. Hybrid power system showing connected devices.

The system proposes an open source, low-cost supervisory control, and data acquisition system using the recently upgraded Arduino IoT cloud platform to monitor and control the prototype of our hybrid system. Current, voltage, temperature, and humidity sensors are connected to the ESP32 microcontroller to read the voltage, current, temperature, and humidity and trigger the relevant LED when the voltage goes below a pre-set value and temperature goes above a stipulated value, as will be explained further in the course of this work. These data can be monitored and controlled from the Arduino cloud dashboards and anywhere using mobile applications. The schematic diagram of the proposed IoT SCADA is shown in Fig. 2.

Fig. 2. Schematic of the Proposed IoT SCADA system

IV. Implementation Methodology

As a proof of concept, a circuit is designed and implemented on a breadboard with three sensors; voltage, current, and temperature and humidity sensors connected to the microcontroller (ESP32) to read the sensor values. The yellow LED is used as the load for which the current flowing through is measured, while the green and the red LED implements the under-voltage logic and over-temperature control logic, respectively. The Wi-Fi compatibility of the microcontroller is leveraged to create the connection with the Arduino IoT Cloud. Dashboards are created in the Cloud to monitor and control the various variables of interest. A mobile application can also be used to control and monitor the variables from anywhere in the world. The connected sensors are the FIDs, and the microcontroller is the RTU. The Arduino IoT Cloud represents the MTU, while the Wi-Fi router at home creates the communication link. The three sensors are the DHT11 temperature and humidity sensor, MH electronics voltage sensor, and ACS 712 Hall-Effect current sensor. The features and properties of these sensors are further discussed and examined below.

A. DHT11 Digital Temperature and Humidity Sensor

This sensor ensures excellent reliability and accuracy by employing an innovative digital signal acquisition technique and temperature and humidity sensing technologies to read the digital signal output. The sensor links to a high-performance 8-bit microcontroller and combines a resistive-type humidity measurement component and a Negative Temperature Coefficient (NTC) measuring component, providing excellent quality, fast response, anti-interference ability, and cost-effectiveness. It has a temperature range of 0 to 50°C, a humidity range of 20% to 90%, 1°C temperature, and 1% humidity precision, respectively. DHT11 comes in two alternative pin layouts, with either four or three pins. The three-pin configuration is used in this work. Vcc is the power supply, Ground is the circuit ground, Data is the serial data output reading the temperature and humidity. The operating voltage of the sensor is between 3 - 5.5 V. A 10kΩ pull-up resistor is used to connect the output pin to the Vcc of the microcontroller. For this work, we are connecting the data pin to GPIO 33 of the ESP32 microcontroller. Fig.3 shows the connection of the sensor to the ESP32 microcontroller.

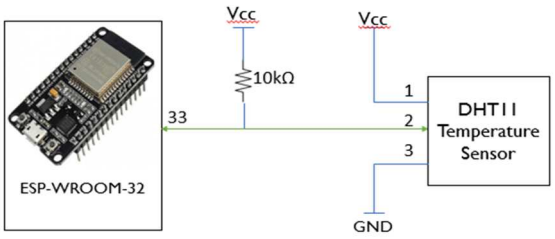

Fig. 3. The connection between ESP32 and DHT11

B. Voltage Sensor Module

This sensor operates by using the voltage divider principle to measure the voltage. It has an embedded series connection of 7.5 kΩ and 30 kΩ resistors to achieve a voltage divider of 5 to 1 for the measurement. The range of operational voltage for the sensor is between 3.3 – 5.0 V, and using a 12-bit ADC can measure voltage in the range of 0 – 25 V DC. Voltage sensors are usually connected in parallel to the voltage source (a 9V AC/DC adapter or a battery) to measure its voltage. ESP32 operates on a 3.3 V input voltage. The sensor is connected directly to measure the voltage as follows: The voltage sensor's pin S is connected to analog pin 32 on the ESP32, and pin – is connected to the ESP32's GND pin, with the sensor's GND and VCC pins connected across positive and negative terminals of the 9 V /5 V buck converter as shown in Fig. 4. The converter is used as the breadboard power supply to provide the needed 5 V needed for the circuit.

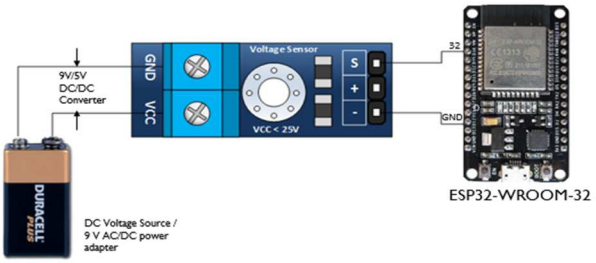

Fig. 4. Interfacing diagram of the voltage sensor with ESP32

C. *ACS712 Hall-Effect Current Sensor*

As the name implies, this sensor works on the principle of Hall-Effect, a phenomenon where a current-carrying conductor placed in a magnetic field generates a voltage perpendicular to both the current and the magnetic field. This generated voltage is used to measure the current. It is a fully integrated, economical, and relatively high precision sensor that employs a non-inversive method of current measurement by measuring the magnetic field created in the wire. ACS712 Hall-Effect sensor can measure both DC and AC by using a low-resistance current conductor. The sensed circuit and the sensing circuit are electrically isolated.

It requires a 5 V DC to power it and comes in three versions of 5A, 20A, and 30A and gives a 2.5 V DC output when no current is detected. Its sensitivity/scale factor ranges from 66 mV/A for 5A, 100 mV/A for 20A, and 185 mV/A for the 30A module. Because the current sensor's signal voltage is 5 V, it is not compatible with direct connection to the ESP32 microcontroller's ADC pins that operate between 0 - 3.3 V. As a result, a pull-down resistors arrangement is implemented to match the current sensor's 5 V signal demand to the ESP32 3.3 V ADC signal capacity, ensuring the accuracy of the measured values [8]. Fig. 5 shows the connection of ACS712 sensor with a step-down resistor configuration to provide the necessary voltage input to the ESP32. Current sensors are always connected in series with the load or source on which the current is measured.

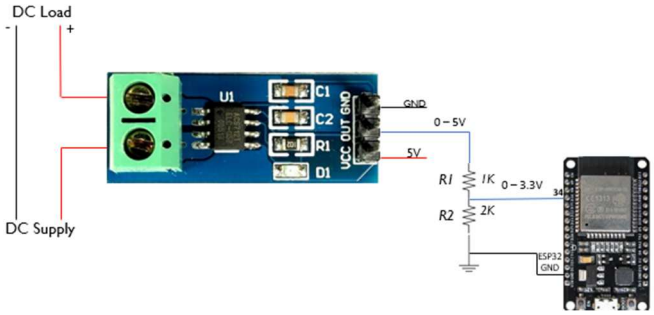

Fig. 5. ACS712 connection to ESP32 microcontroller

The voltage divider equation uses a 1 kΩ and 2 kΩ resistor combination and is expressed as shown in equation (1)

$$V_{ESP} = \frac{R_2}{R_1 + R_2} \times V_{CC} \quad (1)$$

V_{ESP} is the ESP32 voltage, and V_{CC} is the sensor input voltage.

D. *ESP32 – WROOM – 32 Module (RTU)*

This is a powerful microcontroller that supports Wi-Fi, Bluetooth, and Bluetooth Low Energy (BLE) used in diverse applications. Wi-Fi's connecting radius is wider than Bluetooth, which senses the module and connects it to a smartphone with low-energy beacons. The microprocessor uses a dual-core, individually controlled CPU that clocks at an adjustable frequency between 80 MHz to 240 MHz. A co-processor can be employed to save power during low power demanding tasks. It has an infinitesimal sleep mode current (5 µA), making it suitable for battery-powered applications. The Wi-Fi capability of the microcontroller is leveraged to implement the connection to the Arduino IoT Cloud. The Arduino Integrated Development Environment (IDE) in the Cloud is used to write the sketches. The measured data from the sensors are acquired by the board, displayed on the Arduino cloud serial monitor, and sent to the dashboard for monitoring.

E. *Wi-Fi Router (Communication Link)*

Actiontec R3000 FibreOP router is used to create the communication channel between ESP32 (RTU) and the Arduino IoT Cloud MTU). The data transfer rate of this router is 1 Gbps over ethernet and up to 2.3 Gbps over Wi-Fi in dual-band, dual-concurrent 2.4G, 802.11n, and 5G, 802.11ac. Since ESP32 can implement the TCP/IP IEEE 802.11 b/g/n, it is seamlessly connected to the router using the network SSID and password. The router is configured to provide the needed Wi-Fi connection for the implementation of the project. The router's credentials (SSID and the password) provide the needed security against unauthorized access to the system. The login credential of the Arduino IoT Cloud is another layer of security.

F. *Arduino IoT Cloud Platform*

The Arduino community has recently released Arduino IoT Cloud, an IoT platform. The Arduino IoT Cloud features an end-to-end solution that makes creating networked projects simple for creators, IoT enthusiasts, and professionals from inception to delivery. HTTP REST API, MQTT, command-line tools, JavaScript, and WebSocket, are some of the interaction methods supported by the platform. Multiple devices can be linked together, and data can be exchanged in real-time. A simple user interface allows to monitor data from anywhere and execute control when necessary. This platform works with specific Arduino microcontrollers or a third-party board that is compatible with the platform. The third-party boards are ESP32 and ESP8266 microcontroller boards. All cloud-compatible boards come with a cryptographic co-processor with secure hardware-based key storage for enhanced security key storage. The steps involved in creating a project on the Arduino IoT Cloud and all the associated steps, components, and terms are listed below [9, 10]

- **Creating Arduino IoT Cloud Account and Cloud Plan:** Like with every other platform, the first step is to sign up using a functional email address and select a plan.
- **Creating a "Thing":** A "Thing" is the foundation of all Arduino IoT Cloud projects. The term "Thing" refers to a cloud-based virtual item. It stores variables and information about linked devices and networks safely. An online "Thing" editor can generate a Thing for each project you make in the Cloud. A blend of variables, a device, network information, and a sketch will make up an item [9].
- **Connecting Devices to a Network:** Microcontrollers that link to the Cloud are known as devices. Some Arduino microcontrollers and a few third-party boards like the ESP32 used in this work

can also suffice. The network is the Wi-Fi credentials, SSID, and password. The ESP32 used in this work, like other third-party microcontrollers, requires a "secret key" which the Arduino IoT Cloud editor will create when the device (microcontroller) is added along with the network credentials.

- **Creating and Declaring Cloud Variables:** Variables are locations where data are stored. It has a name, holds a value, and are of a particular type. These variables, once declared, are available to the attached microcontroller and the dashboard as well. Its can either be Read and Write or Read Only to. For variables that need to be controlled, we use the Read and Write option, while we use the Read-Only option for variables we are interested in just monitoring.
- **Creating Sketches and Dashboard:** Sketches are the C++ programs written to the microcontroller to execute a defined command. One prominent feature in Arduino IoT Cloud is that when creating a "Thing", a significant portion of the sketch is written automatically when the variables are declared. Sketches are written and edited directly in the Thing editor or using the Arduino Web editor. A dashboard is employed for monitoring and controlling the IoT Cloud Thing. Several widgets are used to build the dashboard and declared variables are linked to the appropriate widget and labeled accordingly. The dashboard can be monitored on the web browser and/or a mobile application on any Android or IOS device to carry out remote monitoring and control.
- **Installing Arduino Create Agent:** Arduino create agent is installed on the PC used for this work. This agent links the PC's USB port on which the microcontroller is connected and the Arduino IoT Cloud. This is necessary because the encryption built into the web browser prevents a website from connecting directly to a PC resource which is essential to our 'Thing' and web editor to work [9].

V. Hardware Prototype Design

The proposed system prototype to demonstrate monitoring and control of a small hybrid power system is designed on a breadboard. The system consists of 9 V AC/DC adapter or (battery), DC/DC converter, current sensor, voltage sensor, temperature and humidity sensor, LEDs (red, green, and yellow), ESP32-WROOM-32 microcontroller, and some pull-down resistors. The 9V is stepped down to approximately 5.5 V by the DC/DC converter. The yellow LED represents the load. A current sensor is connected to analog Pin 34 of ESP32 in series with the yellow LED to measure the current flowing through the LED. The voltage sensor is connected to analog Pin 32 of ESP32 across the output of the DC/DC converter to measure its output voltage (input voltage to the system). The temperature and humidity sensor is connected to analog pin 33 of the ESP32 microcontroller to read the environmental temperature and humidity. The red and green LEDs, with the appropriate "If statements" in the code are used to implement a control logic. They are connected to analog pins 18 and 19 on ESP 32 microcontroller, respectively. When the temperature is greater than 23°C, the red LED comes on. If the measured voltage is less than 4.5 V, the green LED comes on. This control logic is synonymous with what is obtainable in the field, where an air conditioner or an extraction fan regulates the temperature in the shelter or cabin of a BTS site, and the green LED represents the starting relay of a diesel generator that comes ON when voltage drops below a certain threshold (~ 46 V). The ESP32 can be connected either through the USB port or using Over-the-Air (OTA) to upload sketches wirelessly from the Cloud to the board. The prototype is shown in fig.6, and the flowchart for the SCADA system in fig. 7.

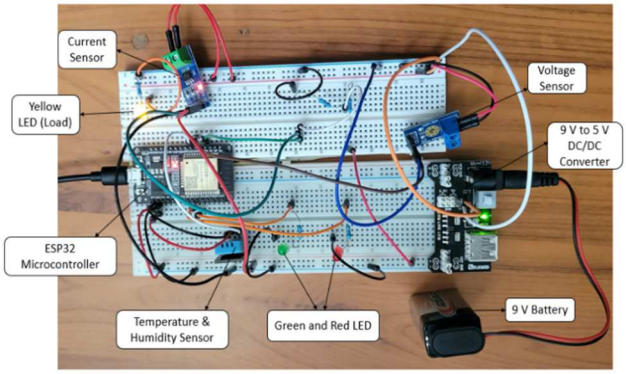

Fig. 6. Experimental circuit setup for the IoT SCADA system prototype

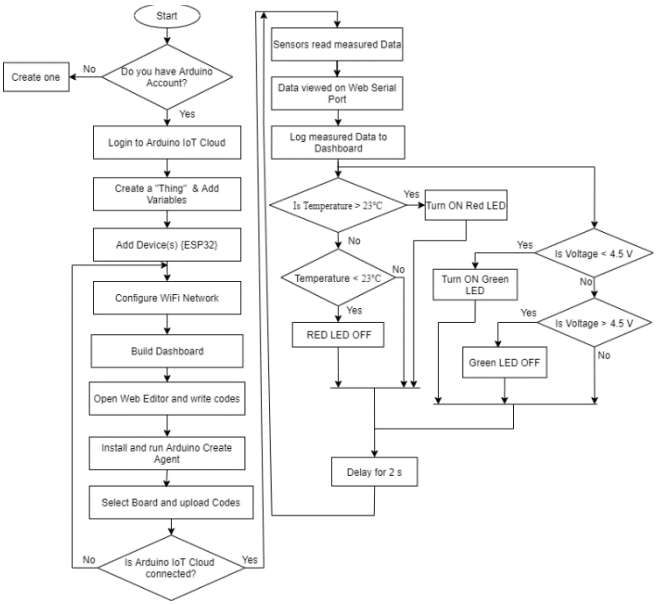

Fig. 7. Flowchart of the IoT SCADA system solution

VI. Testing and Results

The prototype circuit is shown above, and the current, voltage, temperature and humidity were logged every 2 seconds over a period using charts. Instantaneous temperature and humidity were also monitored using a gauge. Gauges are very conspicuous to see the instantaneous values being measured compared to tracing it on a chart. The current measured is that flowing through the load LED (Yellow), and the unit is in milliamp. The voltage measured is the output voltage of the DC/DC converter in Volt. The temperature and humidity are that of the surrounding. Arduino IoT remote mobile application is also installed on my mobile phone where remote monitoring and control can also be carried out from anywhere around the world with the requirement of internet connection and access to the Arduino IoT account login details. The login details ensure only authorized persons have access to the system to monitor and execute any control action. Fig.8. shows the dashboard of the different parameters. The D

represents the days, while H represents the hour. Live is the instantaneous value recorded by the sensors at the very time.

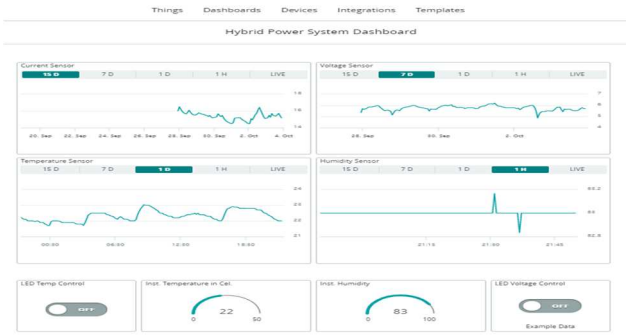

Fig. 8. Dashboard showing the measured parameters at different times.

The historical data stored depends on the Arduino plan chosen. The more expensive plans have a higher data storage capacity compared to the free plan and the less expensive plan.

To demonstrate the control capability of the system, the surrounding temperature is increased to 24°C. Once a higher temperature is recorded beyond our predefined value of 23°C, the red LED comes ON. The LED goes OFF once the temperature goes below the pre-set value. The voltage control can be carried out in the same manner as the temperature. When the voltage drops to 4.5 V, the green LED turns ON and stays ON until the voltage increases above the predefined value.

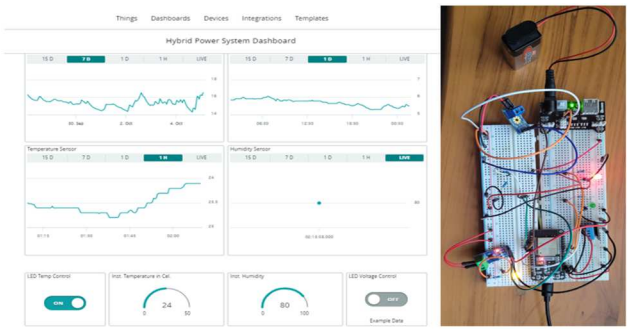

Fig. 9. Temperature control of the system

VII. Discussion

The system is based on the Internet of Things (IoT) configuration. The prototype acts as the plant/process while the sensors are the Field Instrument Devices (FIDs), the ESP32 microcontroller is the Remote Terminal Unit (RTU), Arduino IoT Cloud acts as the Master Terminal Unit (MTU) for creating the necessary Human Machine Interaction (HMI) while the communication channel is the Wi-Fi. The components used for the execution of this project are all low-cost, readily available, and open source. The cost of each component is summarized in table I. In the field, a microcontroller with an option for SIM card like the Arduino MKR boards can be used to provide the necessary internet connection.

TABLE I: LIST OF COMPONENTS AND COST

S/N	Component(s)	Quantity	Price (USD)
1	ESP32 WROOM-32	1	13.85
2	9 V AC/DC power adapter	1	10.34
3	Breadboard Power Module	1	6.40
4	Current Sensor	1	4.76
5	Voltage Sensor	1	4.80
6	Temperature/Humidity Sensor	1	5.20
7	Arduino IoT Cloud plan (Entry) per month	N/A	2.99
8	Miscellaneous (Resistors, Breadboard, LEDs, wires, USB	N/A	40.00
	Total		**88.34 USD**

The dashboard provides the interface to monitor the measured Data both on the PC and mobile device(s). Supervisory control can also be carried out both on the PC dashboard or with the mobile device from anywhere with an internet connection and login details. This provides some form of security from unauthorized personnel.

VIII. Conclusion

The place of monitoring and control in a critical infrastructure like the base transceiver station cannot be over-emphasized. An IoT-based, open source SCADA system is proposed to remedy the limitations of the proprietary SCADA currently in operation. A prototype circuit was used to demonstrate this concept. Three (3) sensors were connected to read the analog values of current, voltage, temperature, and humidity. These data are sent over the Wi-Fi to the Arduino IoT Cloud for monitoring and control through the created dashboard. There is also a mobile application for remote monitoring from anywhere around the world. The cost, though being a prototype, is negligible compared to the proprietary option, and the power consumption is very low as well.

IX. Future Work

The study has been conducted on a prototype circuit. In the future, the authors would like to demonstrate this system on a physical rural telecommunication site to determine the accuracy to which this system can monitor and control the sites' parameters and compare its cost to the proprietary SCADA in use. A database can be incorporated to read the data into a spreadsheet. Also, automatic email and text messages can be sent to relevant authorized personnel to be abreast of the events happening on the site.

References

[1] G. Yadav and K. Paul, "Architecture and Security of SCADA Systems: A Review," International Journal of Critical Infrastructure Protection, p. 100433, 2021/04/08/ 2021.

[2] L. O. Aghenta and M. T. Iqbal, "Development of an IoT Based Open Source SCADA System for PV System Monitoring," in 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), 2019, pp. 1-4.

[3] C. Oton and M. T. Iqbal, "Dynamic Modeling and Simulation of a Stand-alone DC Hybrid Microgrid for a Base Transceiver Station in Nigeria," European Journal of Electrical Engineering and Computer Science, vol. 5, no. 2, pp. 41-49, 2021.

[4] C. Oton and M. T. Iqbal, "Design and Analysis of a Stand-alone DC Hybrid Microgrid for a Rural Base Transceiver Station in Nigeria," in 2020 IEEE Electric Power and Energy Conference (EPEC), 2020, pp. 1-6: IEEE.

[5] A. Soetedjo, Y. I. Nakhoda, A. Lomi, and F. Farhan, "Web-SCADA for monitoring and controlling hybrid Wind-PV power system," Web-SCADA for Monitoring and Controlling Hybrid Wind-PV Power System, vol. 12, no. 2, pp. 305-314, 2014.

[6] C. Vargas-Salgado, J. Aguila-Leon, C. Chiñas-Palacios, and E. Hurtado-Perez, "Low-cost web-based Supervisory Control and Data Acquisition system for a microgrid testbed: A case study in design and implementation for academic and research applications," Heliyon, vol. 5, no. 9, p. e02474, 2019.

[7] I. Allafi and T. Iqbal, "Low-cost SCADA system using arduino and reliance SCADA for a stand-alone photovoltaic system," J. Sol. Energy, vol. 2018, pp. 1-8, 2018.

[8] L. O. Aghenta and M. T. Iqbal, "Low-Cost, Open Source IoT-Based SCADA System Design Using Thinger.IO and ESP32 Thing," Electronics, vol. 8, no. 8, p. 822, 2019.

[9] "DroneBot Workshop," Available online: https://dronebotworkshop.com/arduino-iot-cloud/ (accessed 5 September 2021)

[10] "Arduino.cc," Available online: https://docs.arduino.cc/cloud/iot-cloud/tutorials/iot-cloud-getting-started (accessed 5 September 2021)