Sensors and Materials, Vol. 33, No. 11 (2021) 3903–3915
MYU Tokyo

S & M 2732

Internet-based Remote Setting and Data Acquisition for Fuel Cell

Raydha Zul Fitriani,[1] C. Bambang Dwi Kuncoro,[2] and Yean-Der Kuan[2*]

[1]Graduate Institute of Precision Manufacturing, National Chin-Yi University of Technology,
No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung City 41170, Taiwan
[2]Department of Refrigeration, Air Conditioning and Energy Engineering,
National Chin-Yi University of Technology,
No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung City 41170, Taiwan

(Received May 19, 2021; accepted October 19, 2021)

Keywords: remote setting, Arduino Yun, DAQ, IoT, fuel cell stack

Data acquisition systems (DAQs) have a significant role in acquiring data for convenient analysis. With the spread of the Internet of Things, DAQs with remote access are in high demand in industry. The massive usage of free cloud storage provides an opportunity to build a remote DAQ that provides accessible data around the world. In this paper, a remote setting and DAQ for a fuel cell stack are presented. A prototype was made using Arduino Yun, sensors, and actuators. A data server from Flask Python was built using the Linux side processor of Arduino Yun. Then, the acquired data were sent to Google Drive and saved as Spreadsheets files. A user interface (UI) based on HTML was provided in Google Sites. The UI was used to display the data in table and graph form as well as to set the timing values of the purge valve. Google Apps Script was used to manage data communication with Arduino Yun and also manage the data stored on Google Drive. The proposed system takes 3 s to send data from Arduino to Google Drive, while the time taken to update the purge valve time values is 6 s.

1. Introduction

Data acquisition has become an important fundamental task in modern science and engineering.(1) This can be seen from the number of activities that require data recording against time to observe the progress of systems. Examples of such data are voltages in an electrical system, temperatures, and forces in a mechanical system. Data acquisition systems (DAQs) mostly acquire data from sensors or transducers that produce analog signals.(2) To process these analog signals, it is necessary to convert them to digital signals.(3) A DAQ consists of various components including sensors, signal processors, a computer, communication links, databases, and software. All of these components must work well together for a good DAQ to be built.(4) Microcontroller (MCU) boards such as Arduino have been implemented as low-cost DAQs.(5) González *et al.* used Arduino in the automobile field as a DAQ. The DAQ is made for automobile dynamics applications at frequencies of less than 80 Hz.(6) Mutha *et al.* used Arduino as a real-time DAQ for environmental data useful for agriculture and industry.(7) The previous examples illustrate that MCUs, especially Arduino, have been used in various fields to improve people's lives.

*Corresponding author: e-mail: ydkuan@ncut.edu.tw
https://doi.org/10.18494/SAM.2021.3630

ISSN 0914-4935

The internet has now become a commodity of people, and not only people but also devices are internet users, a concept known as the Internet of Things (IoT).[8] With this development, IoT-based DAQs have emerged. Cruz *et al.* used Arduino Leonardo as a DAQ and combined it with IoT to make a cost-efficient air quality monitoring system that can be accessed through the internet.[9] Then, Mohammad and Shahjahan developed IoT-based supervisory control and data acquisition (SCADA), where the data can be presented in real-time on websites.[10] Combining DAQ with IoT can provide convenience in accessing retrieved data because users can open it via the internet.

In addition to using Arduino, a DAQ can be built with a programmable computer board called Raspberry Pi. Raspberry Pi is a small, inexpensive, powerful, and education-oriented computer board that was introduced in 2012. It operates similarly to a regular PC, requiring a keyboard for command entry, a monitor, and a power supply.[11] Ferencz and Domokos used Raspberry Pi to collect sensor data and an Apache Cassandra database cluster with a web application as an alternative solution for IoT data acquisition and storage systems.[12] Smart boards, such as Arduino and Raspberry Pi, have triggered a revolution in the world of IoT. From another point of view, strategic development for IoT is towards the cloud.[13] Using cloud-based solutions makes it easier to manage big data and information from the IoT world.[14] Melo *et al.* developed a low-cost DAQ for monitoring a photovoltaic system. This DAQ used ESP32 and LoRa to send the data to the Google Cloud platform.[15]

In this paper, a DAQ based on IoT with remote setting capability is presented. The system is made to monitor the data obtained from a fuel cell stack and maintain its performance. The fuel cell stack used is a proton exchange membrane fuel cell stack (PEMFC) with 200 W capacity. The system is built using Arduino Yun as the main component to acquire and log data and adjust the fuel cell system. The logged data are sent through the internet to a free cloud drive called Google Drive. The system is equipped with sensors, actuators, and a user interface (UI) in Google Sites. A UI made using Hypertext Markup Language (HTML) is provided to display the logged data and set up the purge valve of the PEMFC through the internet.

2. Materials and Methods

The DAQ and remote setting system that were made consist of a prototype and UI. A prototype was made to acquire the data from the fuel cell stack, while the UI was created to display the acquired data so that the data can be easily monitored. The system will be discussed further in the following subsection.

2.1 Prototype

The main components of the prototype are Arduino Yun, a temperature sensor (MAX31850K-1 wire thermocouple), a voltage sensor (a series of resistors), a current sensor (ACS712-20A), and a motor driver module (L298N). Supporting components include a purge valve, fans, a DC power supply, cables, jumpers, and screw terminals.

Arduino Yun has two processors, namely, ATmega32U4 as a microcontroller (MCU) and Atheros AR9331 as a microprocessor (MPU). The MPU has a Linux distribution embedded

system (OpenWrt) with the full installation of Python 2.7. The Arduino board has 20 digital I/Os [seven of them can be used as pulse width modulation (PWM) outputs] and 12 analog I/Os. The board also has built-in Ethernet and WiFi.

A MAX31850K temperature sensor is used, which is a breakout board manufactured by Adafruit. The board includes a chip and a 3.3 V regulator. To measure the temperature, the board requires a thermocouple. In this prototype, a type K thermocouple is used. An ACS712 current sensor with a capacity of 20 A is used. This is because the current generated by the fuel cell stack can reach 10 A. Thus, the sensor can still measure the largest current generated by the fuel cell.

For the voltage sensor, this prototype uses seven resistors in series whose total resistance is 137 kΩ. The resistance of each resistor in order of appearance in the circuit is given in Table 1. In addition, the resistor circuit is shown in Fig. 1.

For the voltage sensor values to be read by Arduino, the order of the resistors in the circuit is very important because Arduino can only receive a maximum signal of 5 V. To determine the resistances that should be used, the total resistance is first calculated using Eqs. (1) and (2). Both equations are also used to determine the voltage reading point via Arduino. The amount of resistance used is designed so as not to have a load of more than 1 mA on the circuit and also the voltage signal transmitted to Arduino is not more than 5 V.

$$CV = \frac{R_{inAr}}{R_{total}} = \frac{R5 + R6 + R7}{R1 + R2 + R3 + R4 + R5 + R6 + R7} \tag{1}$$

$$V_{inAr} = CV \times V_{fc} \tag{2}$$

Equation (1) is used to ensure that V_{inAr} in Eq. (2) has a value of 5 V by determining the multiplier coefficient (CV). To obtain CV, R_{inAr} is divided by the total resistance. Then, CV is multiplied by the fuel cell voltage (V_{fc}). In this study, the open voltage of the fuel cell can reach 38 V, so the value of V_{fc} used is 40 V for safety. If the calculated V_{inAr} is close or equal to 5 V, it indicates that the resistance is suitable for handling fuel cell voltages. As shown in Fig. 1, the circuit is connected to Arduino via point D.

Table 1
Resistances of resistors.

Resistor	Resistance (kΩ)
R1	27
R2	33
R3	27
R4	33
R5	1
R6	1
R7	15
Total	137

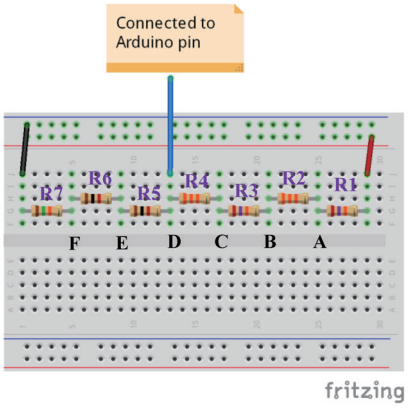


Fig. 1. (Color online) Arrangement of resistors.

Before the voltage sensor is used, the calibration process is carried out manually. The tools used for the calibration are a voltmeter (multimeter), voltage sensor, power supply, the Arduino board, and a computer. A schematic of the arrangement of the tools used for calibration is shown in Fig. 2.

In the calibration, the multimeter is a reference tool used to compare the measured values of the voltage sensors. Calibration is carried out by supplying a voltage of 0 to 33 V from the power supply to the voltage sensor. When the voltage is applied, the voltage across the sensor is measured using the multimeter and then recorded. At the same time, Arduino also measures the voltage on the sensor as an integer value ranging from 0 to 1023, which in this paper is called the raw value. Both values are recorded in Microsoft Excel and plotted on a graph. Then, using the trendline feature of Excel, the formula for calculating the voltage value measured by Arduino is obtained. The formula is then applied to measure the voltage with Arduino. If the voltage values from the calculation and the measurement by the multimeter have a very large difference, the previous calibration steps are repeated until the calculated voltage value is close or equal to the measured voltage value. A flowchart of the calibration steps is shown in Fig. 3.

To calculate the current value that Arduino obtained from the current sensor, a formula based on the coefficients obtained from the sensor datasheet is used. In contrast, the temperature sensor is calibrated using a library that is already available on Arduino. Every time Arduino is turned on, the temperature sensor is calibrated immediately using the library.

Before all the previously mentioned components are assembled together, both the main components and the supporting components are tested to ensure that each component works properly. After that, the assembled components are tested to check that they work properly when turned on at the same time. Figure 4 shows a schematic of the prototype circuit.

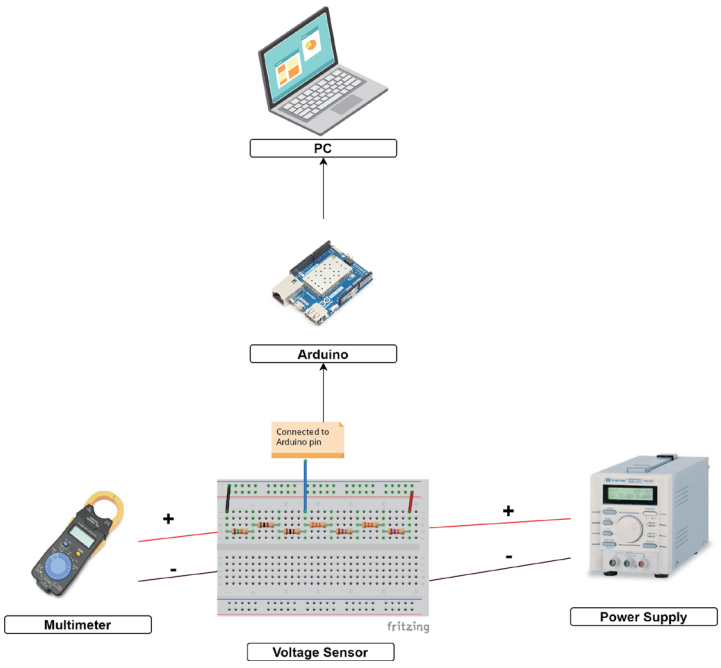


Fig. 2. (Color online) Schematic of arrangement of calibration tools.

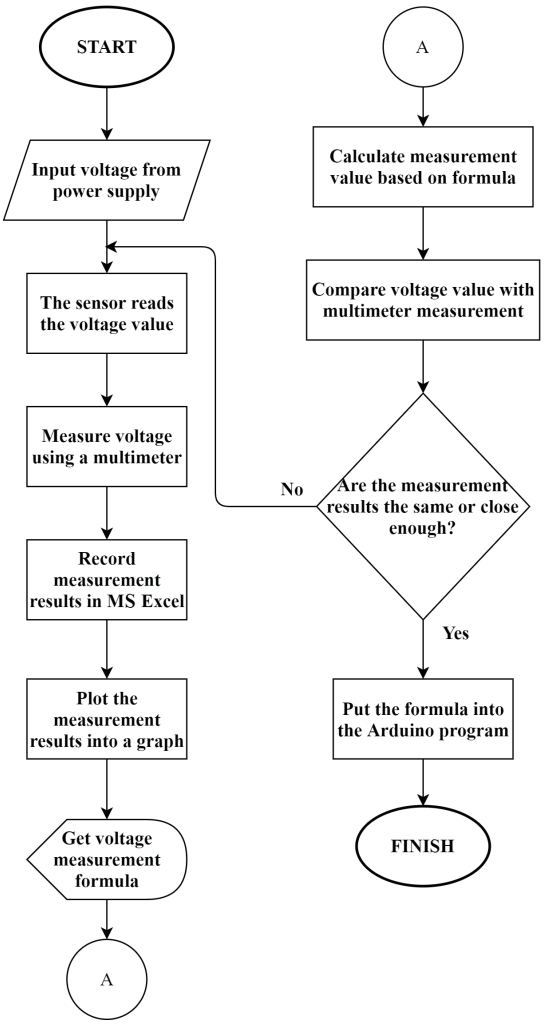


Fig. 3. Flowchart of voltage sensor calibration steps.

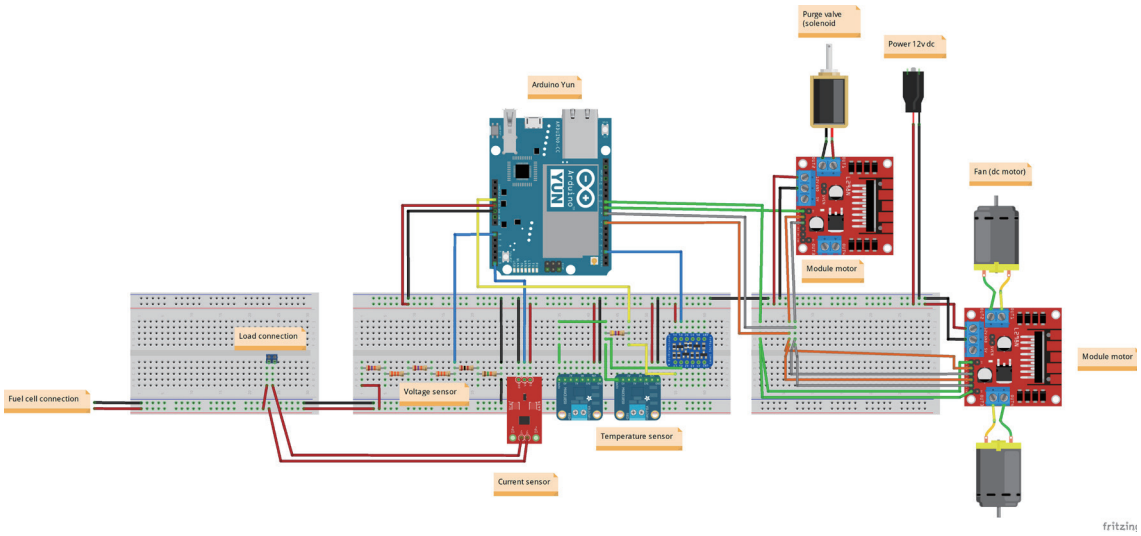


Fig. 4. (Color online) Schematic of the prototype circuit.

As shown in Fig. 4, each sensor and actuator are connected to Arduino. The MAX3850K sensor is connected to a level shifter to protect the sensor board from voltage differences with Arduino, and each data line from each MAX3850K sensor board is connected to a 4.7 kΩ resistor and a 3 V connection from the level shifter. Then, the shared data line is connected to digital pin 2 (D2). The ACS712 sensor is connected to analog pin 5 (A5) and the output of the sensor is connected to the load of the fuel cell. The voltage sensor, which comprises a series of resistors, is connected to Arduino pin A0, while the output is connected to the fuel cell load. In addition to sensors, there are also actuators in the circuit. There are two motor driver modules: one has an attached solenoid valve as the purge valve in the fuel cell and the other has two fans attached to drive the fuel cell fan. The motor driver module is supplied with a 12 V power supply. The motor driver module for the purge valve is connected to Arduino pin D9 and the motor driver module for the fans is connected to pin D10. The two pins used for the motor driver module are PWM-type pins. To build the prototype, screw terminals are used to connect each cable and wire.

2.2 Arduino Sketches

Arduino Yun is programmed using Arduino Integrated Development Environment (IDE) software. The language of this software is a simplified form of C++. Sketches is the name of a program written using the Arduino IDE software. In this system, a written "sketch" contains libraries, variables, and commands allowing Arduino to manage sensors and actuators, as well as enable communication between the MCU and MPU.

First, the sketch initializes all the libraries and variables used. Then, a bridge to activate the communication path between Linux and the MCU is turned on. After that, Arduino reads the values of the interval time and purge time of the purge valve on the SD card. In the loop mode, Arduino applies these values to the motor driver to drive the purge valve. Then, all the sensors begin to acquire data and Arduino records each data into the SD card. The data are saved as a CSV file. When there is a change in the value of the interval time or purge time, a specific command prepared in the sketch accepts these values and Arduino applies the values to the purge valve. In addition, the values are also saved to the SD card.

The data are logged periodically every hour. Data files are named on the basis of the date and time the data are taken. Temperature, voltage, and current data are recorded. For each recorded data, the time of collection is also recorded with the data.

2.3 UI

The UI is created using HTML. The scripting language used is JavaScript. The UI is embedded in Google Sites and can be accessed via the internet. The designed UI has three tabs, a calendar, and a button to select files. The calendar is used to select a file and sheet name that users want to display. The desired sheet is selected on the basis of the selected hour in the calendar. The three tabs are the Data Table tab, Data Plot-Graph tab, and Setting Tab. The design of the UI is shown in Fig. 5.

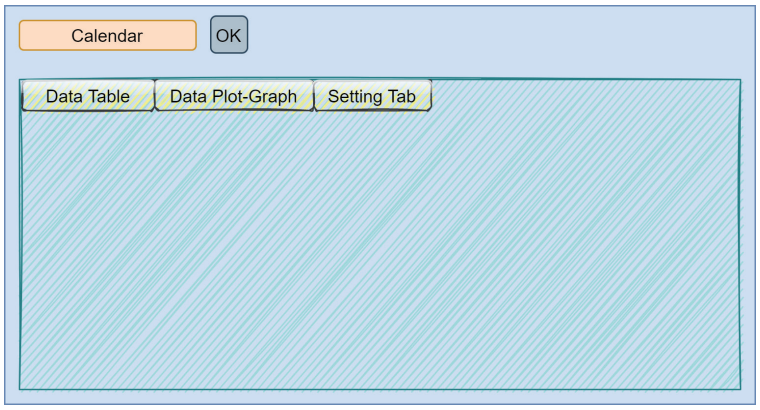


Fig. 5. (Color online) Design of UI.

The Data Table tab is used to display data in the form of a table. The columns of the table show the timestamp, temperature, current, voltage, and power. The Data Plot-Graph tab displays historical graphs for each type of data. To display graphics in HTML, a library called Flot is used. In Flot, there are many examples of suitable graphs and this library uses the JavaScript language that matches the UI created. To use the Flot graphs, it is only necessary to modify the given example and use the appropriate library. The Setting Tab is used to change the interval time and purge time of the purge valve. In this tab, there are two inputs where the values of the interval time and purge time can be entered.

2.4 Data management

All data from the sensor are recorded by Arduino into the SD card. Then the data are sent to be stored on Google Drive in the form of a Spreadsheets file. When the data are sent to Google Drive, each row of recorded data is sent one by one. A Python-based data server is created to transmit the data. The data server is built in an MPU that runs Linux and has a full Python 2.7 installation. A web framework for Python called Flask is used to build the data server. Data are sent using Hypertext Transfer Protocol (HTTP) communication. On Flask, HTTP tasks are executed using a library called Requests. In this library, the HTTP methods used are GET and POST. HTTP POST is used to send data to Google Drive, while HTTP GET is used to retrieve data from Google Drive to set the values on the purge valve.

On the internet side, data are managed by Google Apps Script, hereafter referred to as Apps Script. Apps Script receives the data sent by Flask and then manages the data storage on Google Drive. Apps Script was programmed in advance to save the data in the form of a Spreadsheets file and arrange the data according to the time retrieved. Apps Script also manages the data displayed on the UI; the data displayed are the data selected by the user through the calendar. A flowchart of the two steps described above is shown in Fig. 6.

To set the purge valve timing value, values entered in the Setting Tab of UI are saved in a Spreadsheets file; this step is also performed via Apps Script. Then, when Python detects a new value, it downloads the value in CSV format. The downloaded value is then sent to the MCU on Arduino via the communication socket and applied as the purge valve time value.

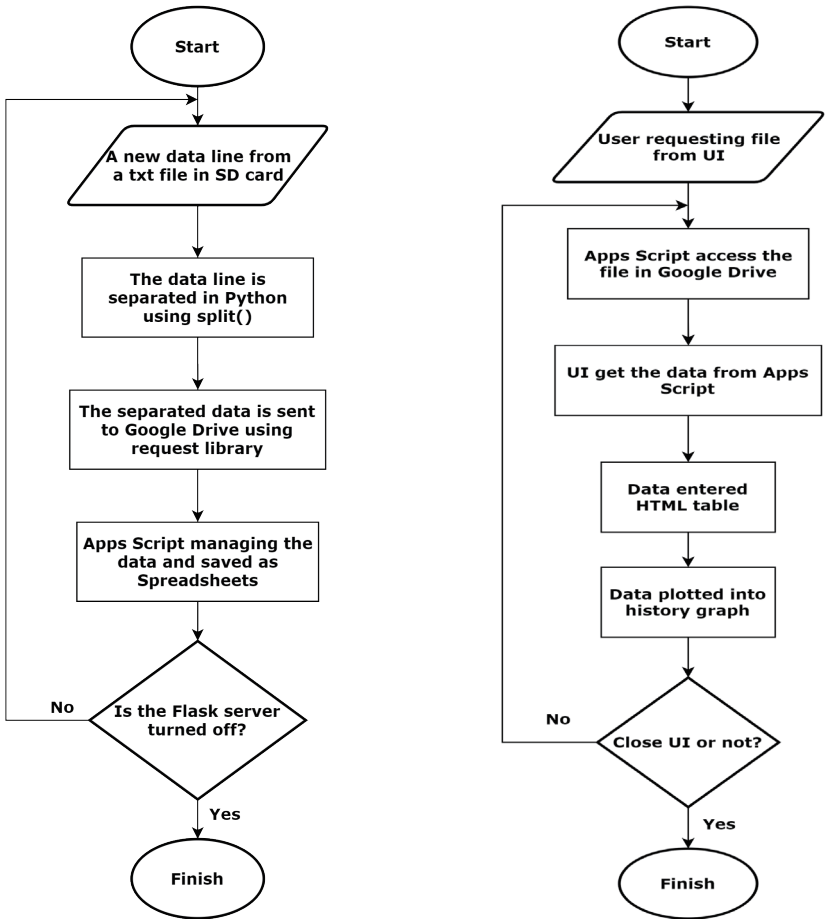


Fig. 6. Data management flowchart. (a) Sending data to Google Drive and (b) displaying data on UI.

3. Implementation and Results

After assembly and testing, the prototype is then used to retrieve data from the fuel cell stack. The results of the assembled prototype are shown in Fig. 7. Figure 8 shows the setup of the experiment conducted using the prototype to retrieve the data from the fuel cell stack.

The experiment is carried out using a fuel cell stack with a capacity of 200 W. The data taken are temperature, current, and voltage. The data are taken at the same measurement point as the fuel cell test machine, which is used to conduct experiments on fuel cells in this lab. During the experiment, the data server actively sends data to Google Drive. The command prompt of Flask sending the data to Google Drive is shown in Fig. 9.

The orange rectangle in Fig. 9 shows the feedback messages indicating that the data are successfully sent to Google Drive. The delay time experienced is 3 s. Then the data are transferred into the appropriate Spreadsheets file by Apps Script. With the data saved in the Spreadsheets files, it is easier to access the data. In addition, there is a UI to display the data via

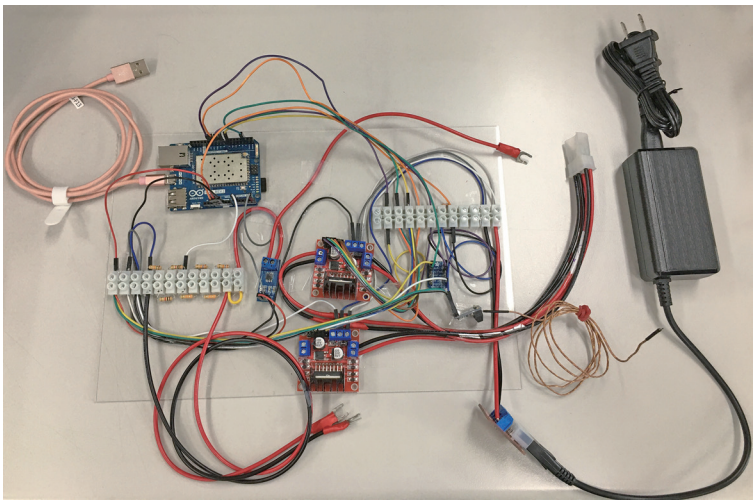


Fig. 7. (Color online) Assembled prototype.

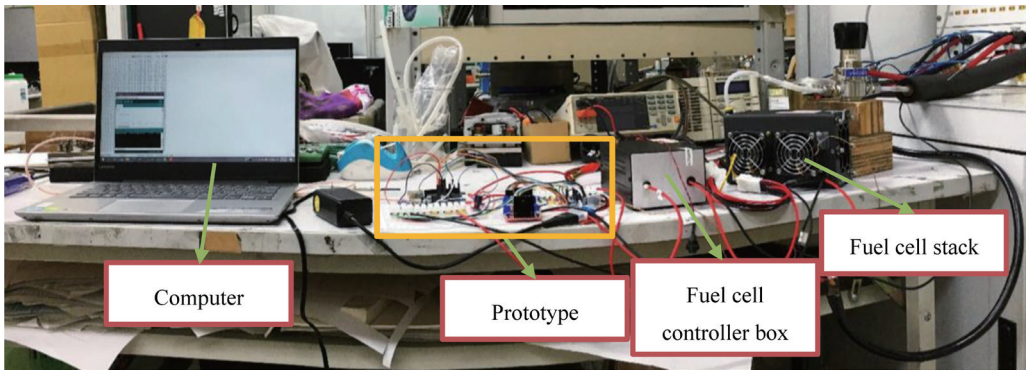


Fig. 8. (Color online) Setup of experiment on fuel cell stack with the prototype.

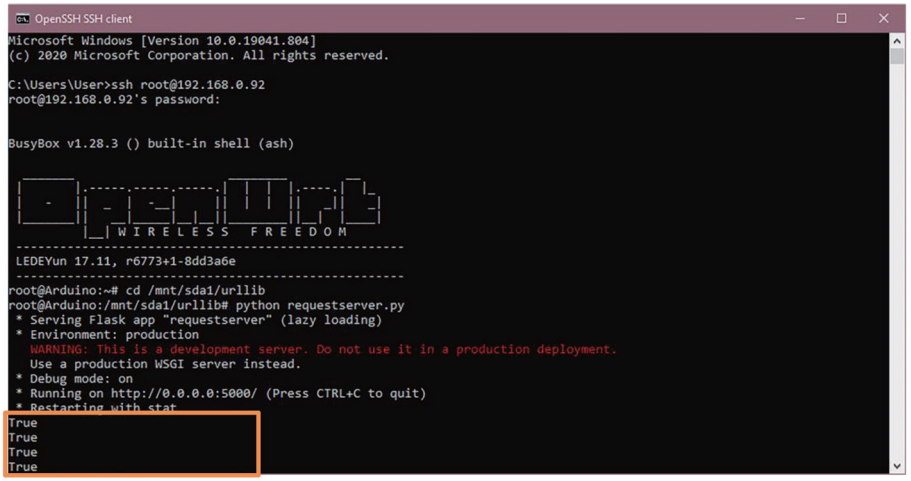


Fig. 9. (Color online) Command prompt of Flask Python.

the internet. The Google Sites homepage used as the UI is shown in Fig. 10. As previously explained, the data in the UI can be displayed in the form of tables and graphs, as shown in Figs. 11 and 12, respectively.

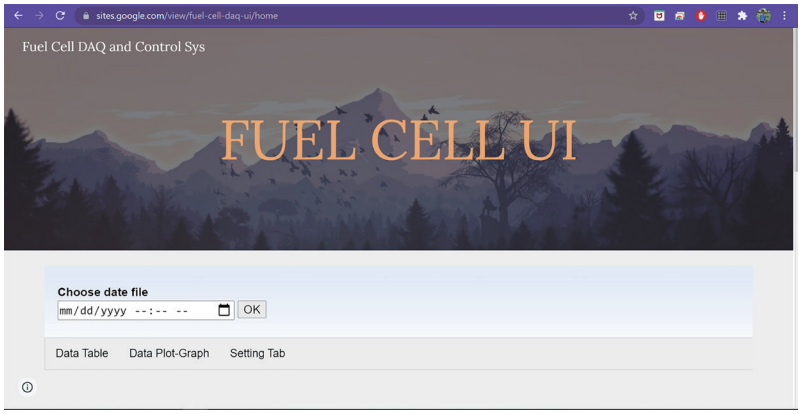


Fig. 10. (Color online) Homepage of Google Sites.

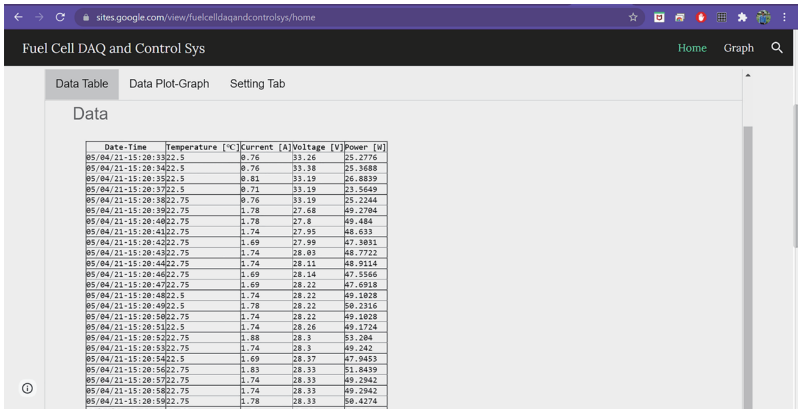


Fig. 11. (Color online) Data in table form.

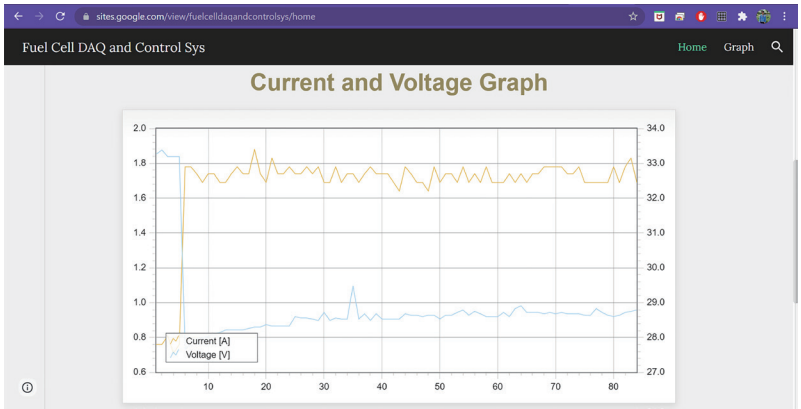


Fig. 12. (Color online) Data in graph form.

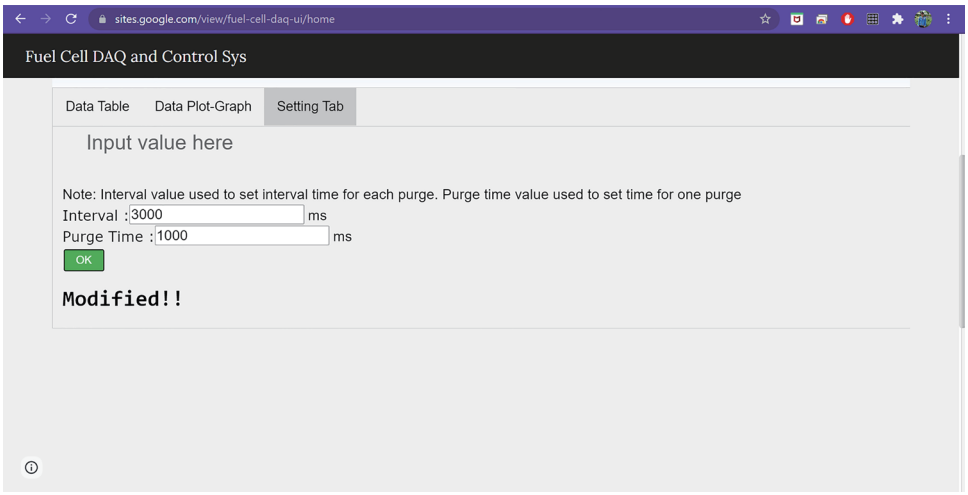
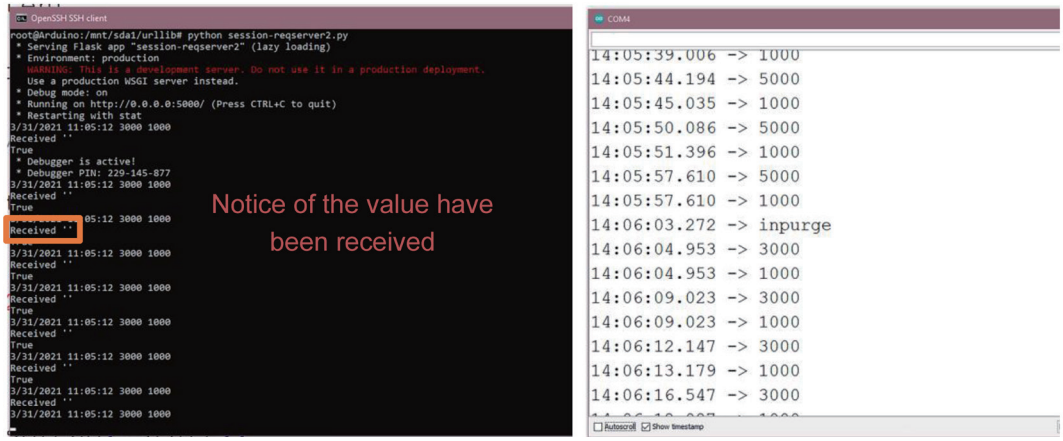


Fig. 13. (Color online) Setting Tab in the UI.



(a) (b)

Fig. 14. (Color online) (a) Flask command prompt with feedback message and (b) Arduino serial monitor.

The plotted graph shows historical data from the selected file. There are three types of graphs, namely, temperature graphs, current and voltage graphs, and power graphs. Graphs for historical data are plotted in the Data Plot-Graph tab.

The Setting Tab is used to enter the purge valve time values as shown in Fig. 13. To set the time, the user must enter the desired time value, then click the "OK" button. After that, the values are saved as a Spreadsheets file. Flask obtains these values and sends them to Arduino. Arduino then changes the interval time and purge time of the purge valve. Figure 14 shows a message in the command prompt acknowledging that the values have been accepted by Arduino, and the Arduino serial monitor also shows that the time value has been changed in accordance with the given value. Arduino takes 6 s to update the values of the purge valve.

4. Conclusions

A DAQ with remote setting capability for a fuel cell stack was made using Arduino Yun. A prototype was made using Arduino Yun, a temperature sensor, a current sensor, and a voltage sensor. The system was also equipped with a remote setting feature to change the time values of the purge valve. The settings are changed through a UI that was also made along with the prototype. The UI was created using HTML format. The data are sent to Google Drive via Flask, a Python web framework. In Google Drive, data are organized via Apps Script, so data can be stored according to the time of retrieval. It takes 3 s to send data from Arduino to Google Drive, and Arduino takes 6 s to update the time values of the purge valve. As the data are saved in Google Drive and can be accessed through the internet, the data can be provided anytime and anywhere for every user. In addition, the remote setting feature used to modify the interval and purge time of the purge valve enables the system to be IoT-based.

Acknowledgments

This research was supported by the Ministry of Science and Technology, Taiwan, under project number 110-2221-E-167-013-MY2 and the Electronics Cooling and Fuel Cell Laboratory of Refrigeration, Air Conditioning and Energy Engineering Department, National Chin-Yi University of Technology.

References

1 S. Rei: Balkan Reg. Conf. Eng. Bus. Educat. **2** (2017) 130. https://doi.org/10.1515/cplbu-2017-0018
2 M. Abdallah and O. Elkeelany: ICC (2009) 240. https://doi.org/10.1109/ICC.2009.24
3 Ichwana, I. S. Nasution, S. Sundari, and N. Rifky: IOP Conf. Series: Earth and Environmental Science **515** (2020) 012011. https://doi.org/10.1088/1755-1315/515/1/012011
4 P. Sarma, H. K. Singh, and T. Bezboruah: IJCSE **6** (2018) 539. https://doi.org/10.26438/ijcse/v6i6.539542
5 R. Naveenkumar and Dr. P. Krisnha: IJSR **2** (2013) 366. https://www.ijsr.net/archive/v2i2/IJSRON2013409.pdf
6 A. González, J. L. Olazagotia, and J. Vinolas: Sensors **18** (2018) 366. https://doi.org/10.3390/s18020366
7 V. R. Mutha, N. Kumar, and P. Pareek: 2016 IEEE 1st Int. Conf. Power Electronics, Intelligent Control and Energy Systems (ICPEICES) (IEEE, Delhi, 2016) 1–4. https://doi.org/10.1109/ICPEICES.2016.7853337
8 S. M. Huq, M. A. Rahman, and S. M. Saleh: 2017 20th Int. Conf. Computer and Information Technology (ICCIT) (IEEE, Dhaka, 2017) 1–4. https://doi.org/10.1109/ICCITECHN.2017.8281837
9 L. A. A. Cruz, M. T. T. Griño, T. M. V. Tungol, and J. T. Bautista: I. J. Eng. Manuf. **9** (2019) 1. https://doi.org/10.5815/ijem.2019.03.01
10 N. Mohammad and Md. Shahjahan: Proc. 2019 5th Int. Conf. Advances in Electrical Engineering (ICAEE) (IEEE, Dhaka, 2019) 306–311. https://doi.org/10.1109/ICAEE48663.2019.8975695
11 M. Maksimović, V. Vujović, N. Davidović, V. Milošević, and B. Perišić: Proc. 1st Int. Conf. Electrical, Electronic and Computing Engineering IcETRAN 2014 (IEEE, Vrnjačka Banja, 2014) 1–6.
12 K. Ferencz and J. Domokos: 2018 Int. IEEE Conf. and Workshop in Óbuda on Electrical and Power Engineering (CANDO-EPE) (IEEE, Budapest, 2018) 000143–000146. https://doi.org/10.1109/CANDO-EPE.2018.8601139
13 D. Bruneo, S. Distefano, F. Longo, G. Merlino, A. Puliafito, and A. Zaia: 2017 Global Internet of Things Summit (GIoTS) (IEEE, Geneva, 2017) 1–6. https://doi.org/10.1109/GIOTS.2017.8016263
14 R. Z. Fitriani and Y.-D. Kuan: The 31st National Conf. Combustion and Energy (NFU, Yunlin, 2021) 452–459.
15 G. C. G. d. Melo, I. C. Torres, I. B. Q. d. Araújo, D. B. Brito, and E. d. A. Barboza: Sensors **21** (2021) 3293. https://doi.org/10.3390/s21093293

About the Authors

Raydha Zul Fitriani received her B.S. degree from Politeknik Negeri Bandung, Indonesia, in 2019 and her M.S. degree from National Chin-Yi University of Technology (NCUT), Taiwan, in 2021. Currently, she is pursuing her Ph.D. degree at the Graduate Institute of Precision Manufacturing, NCUT. Her current research interests are in fuel cells and IoT. (slb011007@student.ncut.edu.tw)

C. Bambang Dwi Kuncoro received his diploma in electrical engineering from Polytechnic of Bandung Institute of Technology (ITB), Indonesia, in 1993 and his bachelor's and master's degrees in electrical engineering from ITB in 1999 and 2002, respectively. He received his Ph.D. degree in precision manufacturing from the Graduate Institute of Precision Manufacturing, National Chin-Yi University of Technology (NCUT), Taiwan, in 2020. Since 2020, he has been a professor at the Department of Refrigeration, Air Conditioning and Energy Engineering, NCUT. His research interests include sensor integration, embedded systems, control and monitoring applications, wireless sensor networks, wireless power transfer, and renewable energy. (bkuncoro@ncut.edu.tw)

Yean-Der Kuan is a distinguished professor and former chairman (2013/02–2019/01) of the Department of Refrigeration, Air Conditioning and Energy Engineering, National Chin-Yi University of Technology, Taichung City, Taiwan. He received his Ph.D. degree from the Department of Mechanical and Aerospace Engineering, University of Missouri, USA, in 2000. Currently, he is the director of the Taiwan Society of Heating, Refrigeration and Air Conditioning, the director of the Taiwan Energy Association, the director of the Taiwan Association for Hydrogen Energy and Fuel Cells, and a member of the American Society of Heating, Refrigeration, and Air Conditioning. His research interests include the fields of energy saving and renewable energies, and air-conditioning components and systems. (ydkuan@ncut.edu.tw)