JTE HCMUTE
ISSN: 2615-9740

JOURNAL OF TECHNICAL EDUCATION SCIENCE
Ho Chi Minh City University of Technology and Education
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn

Develop an RS-485 Protocol for Arduino Boards Applied To Networked Real Time Control Systems

Dang Long Tran[1*], Truong Hoa Binh Nguyen[1], Nam Hoa Ho[1], Duy Anh Nguyen[1], Van Danh Tran[2], Minh Nhat Nguyen[2], Duc Chanh Tin Doan[2]

[1]*Ho Chi Minh City University of Technology, Vietnam National University Ho Chi Minh City, Vietnam*
[2]*Institute for Nanotechnology, Vietnam National University Ho Chi Minh City, Vietnam*

[*]Corresponding author. *Email: trandanglong@hcmut.edu.vn*

ARTICLE INFO

Received: 12/08/2023
Revised: 25/09/2023
Accepted: 29/11/2023
Published: 28/08/2024

KEYWORDS

Arduino board;
RS-485 protocol;
Master-Slave network;
Supermaneuverable EV;
Saltwater intrusion.

ABSTRACT

The Arduino microprocessor boards such as Mega 2560, UNO R3, Leonardo, Micro, and Nano are simple and low-cost tools for real-time measurement and control applications. These Arduino boards cannot be used in distributed systems because they lack the networking capabilities to transfer data across units. In this study, an RS-485 protocol for Arduino boards that operate in Master-Slave networks was developed. Network operations could be carried out independently on the main thread program, and devices in the network could react quickly to information received. This was made possible by the asynchronous serial communication feature and a high-speed timer provided in Arduino boards. The networks designed in this study were applied to an electric vehicle model with all-wheel drive and all-wheel steering capabilities for supermaneuverability as well as a saltwater intrusion early warning system installed in a river entry. The results showed that highly reliable and stable network operations could be achieved, thus extending the usage of popular Arduino boards for networked real-time applications.

Doi: https://doi.org/10.54644/jte.2024.1445

1. Introduction

Arduino microprocessor boards are a low-cost, high performance development platform widely used in embedded system applications [1]-[7]. The platform supports basic communication protocols such as UART, I^{2}C, and SPI. While the I^{2}C allows multiple devices to connect and form a local network with a maximum transmission distance of less than one meter, the UART and SPI only allow communication between two devices. Consequently, Arduino boards are not suitable for distributed data acquisition and control systems in which sensors and actuators exchange data with controllers over long distances via network protocols.

In comparison to UART, I^{2}C, and SPI, the RS-485 protocol supports high speed serial data transmission over much longer distances, up to thousands of meters, at speeds of up to 1 mega bits per second. The RS-485 also supports Master-Slave communication model where one device, referred to as the Master, controls and directs the actions of up to 31 subordinate devices, known as Slaves, allowing for increased flexibility and scalability. The RS-485 also has high reliability and noise immunity by means of using a twisted pair cable, ensuring stable data transmission even in noisy environments. These advantages make the RS-485 particularly useful in IoT and industrial applications where connecting to remote devices is necessary [8].

It is clear that with RS-485 networking functionality, an Arduino-based device can connect and communicate with multiple devices on a single network, expanding the application capabilities and enhancing the Arduino platform's versatility. To make Arduino boards running with the RS-485 protocol by firmware is less expensive than using higher performance microprocessor boards having an integrated CAN protocol which is handled by hardware. Furthermore, whereas the CAN protocol specifies a fixed data frame length, the RS-485 protocol is completely customizable.

JTE
HCMUTE
ISSN: 2615-9740

JOURNAL OF TECHNICAL EDUCATION SCIENCE
Ho Chi Minh City University of Technology and Education
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn

This study is aimed to develop an RS-485 protocol for commonly used Arduino boards such as Mega 2560, UNO R3, Micro, and Nano for real-time control over the Master-Slave network architecture. By using an external UART/RS-485 interfacing hardware in conjunction with the UART communication and a high speed timer provided in Arduino boards, network operations of each Arduino board including serial data transmission, message frame generation and interpretation, network communication control can be carried out independently of the main thread program, and devices in the network can react quickly to information received.

The paper is organized as follows. Section 2 describes the overall design of an RS-485 Master-Slave network using Arduino boards including network configuration, message frame format, and data flows. Section 3 introduces algorithms for message transmission and receive with fast response. Section 4 demonstrates the use of the newly developed RS-485 network in automotive and environmental domains. Finally, in the Conclusion, key findings are summarized.

2. Overall design of RS-485 Master-Slave network using Arduino boards

To built an RS-485 protocol for Arduino boards, network configuration, message frame format, and data flows have to be first determined.

2.1. Network configuration

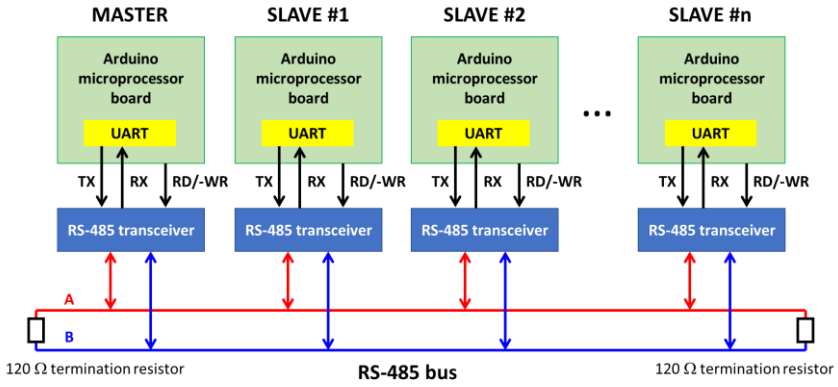

Figure 1. *The schematic diagram of an RS-485 network using Arduino microprocessor boards having (n+1) devices, including one Master and n Slaves.*

An RS-485 Master-Slave network consists of a Master device and several Slave devices. All devices are physically connected via a single communication called a bus which is a pair of wires (one called the "A" and the other called "B") with 120 Ω resistors at each end to minimize signal reflections, preventing signal degradation. Data bits are broadcasted by one of the devices to all the others in terms of balanced differential signals in the "A" and "B" wires. The Master controls the communication by sending data packets (also called message) to the Slaves, and the Slaves respond as needed. Generally, the Master controls, acquires information and send commands to the Slave devices, and the Slaves carry out tasks assigned by the Master. For addressing, the Master and the Slaves have different ID numbers.

To build this type of network for Arduino boards, the UART function provided by the Arduino board is used for serial data transmission and receive via the TX and RX pins (Figure 1). Each Arduino board needs an external RS-485 transceiver as its UART/RS-485 interface to access the network. Data flow direction between the Arduino board and the network is selected by a Read/Write control signal. With a read command (e.g. Read/Write = "1"), the Arduino board listens to the network and receives all data bits broadcasted. With a write command (e.g. Read/Write ="0"), the Arduino board is the source of data broadcasting. To avoid data collision, only one source of data broadcasting is allowed at a time.

2.2. Message frame format

As presented in Figure 2, each message sent over the network consists of three parts: a Header, a Payload, and a Trailer. In this study, the Header field has four bytes, including a pre-defined character as Start-of-Frame (SOF), the source ID number, the destination ID number, and the number of Payload bytes (so-called the Data Length (DL)). The Payload field contains application data that are needed to

be sent to the destination. The Trailer field is designed with four bytes, including three reserved bytes and a pre-defined character as End-of-Frame (EOF). The total frame length is N+8 bytes.

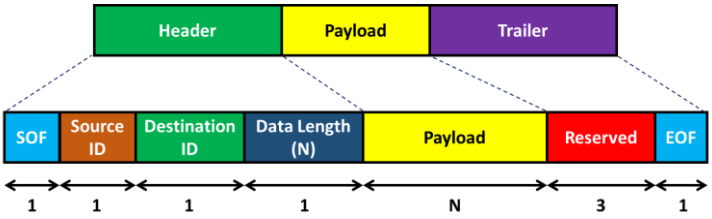

Figure 2. *The message frame format of the RS-485 Master-Slave network proposed in this study. Numbers indicate the number of bytes in each field. The total frame length is N+8 bytes. SOF-Start of Frame. EOF-End of Frame.*

Because a message is transmitted as a bit stream, the user-defined SOF and EOF characters are used to recognize a complete message frame from which the contents of all fields can be extracted.

The Master initiates a communication by sending a message to one of the Slaves. In this message, the source ID number is the Master ID number, and the destination ID number is the ID number of the Slave that needs to subsequentially response. Therefore, all the Slaves simultaneously receive the message sent by the Master but only one Slave that is specified by the destination ID responses while all the others ignore the message.

2.3. Data flows

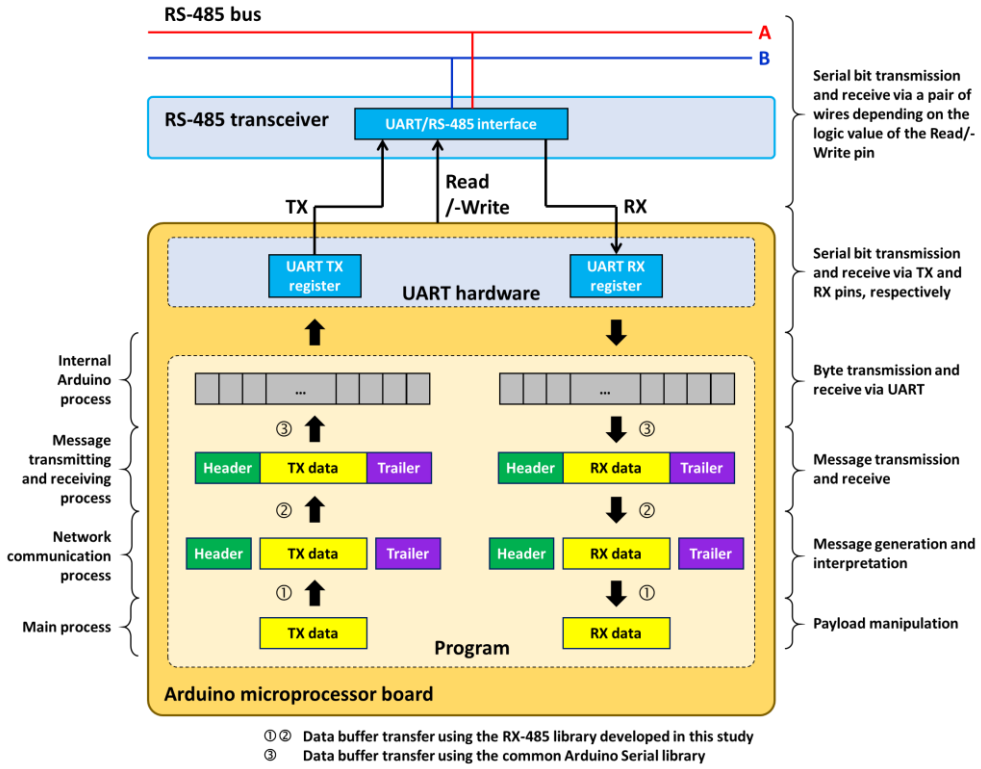

Figure 3. *Data flows in an Arduino board connected to the RS-485 network. Message generation and interpretation can be varied correspondingly to specific applications. Message transmission and receive are developed in this study to achieve fast operation.*

In each Arduino board, application data that needs to be sent through the network is stored in a TX buffer, and application data received from the network is stored in an RX buffer. These two buffers are manipulated in the main process of the Arduino program. To broadcast application data to the network, the TX buffer content is considered as a payload and is transferred from the main process to a network communication process, where a header and a trailer with appropriate contents are attached to form a complete message frame. The produced message frame is then transferred to a message transmitting and

receiving process running in a timer interrupt service routine, where this message waits to be transferred to a serial write buffer at a desired time. Each byte of the serial write buffer is sequentially copied to the empty UART TX shift register of the Arduino microprocessor by the internal Arduino process. Once the TX shift register gets a new byte, its bits are automatically emitted out of the TX pin of the Arduino board as a bit stream until it is empty again.

Depending on the Read/Write control signal, each Arduino board can be in either the default Listening mode or the Sending mode. Prior to the bit stream transmission, the Read/Write control signal is set to the Write value (e.g. "0") to dominate the communication line for broadcasting. Once transmission is completed, the Read/Write control signal should be set to the Read value (e.g. "1") as soon as possible, quickly releasing the communication line.

In the listening mode, the bit stream sent over the network is automatically captured as bytes and stored in the UART RX shift register by the Arduino microprocessor, subsequentially transferred to a serial read buffer by the internal Arduino process. The message transmitting and receiving process have to regularly check the serial read buffer for new received bytes and searchs for a complete message frame basing on the pre-defined SOF and EOF characters. Once detected, the Header, the Payload, and the Trailer are segmented from the frame. The destination ID number is checked to defined whether the received Payload is safely neglected or not. If the ID number of the destination is identical to that of the receiver, message parts are transferred to the network communication process, and then only the Payload is transferred to the main process for further manipulations.

The data flows mentioned above is illustrated in Figure 3. The network communication process can be varied depending on specific applications. Data transfers between the message transmitting and receiving process and the internal Arduino process are carried out using the common Arduino Serial library. Meanwhile, a message transmitting and receiving process need to be developed in this study so that a fast response of the destination device can be achieved, allowing high rates of data packet sent through the network.

3. Algorithm for message transmission and receive with fast response

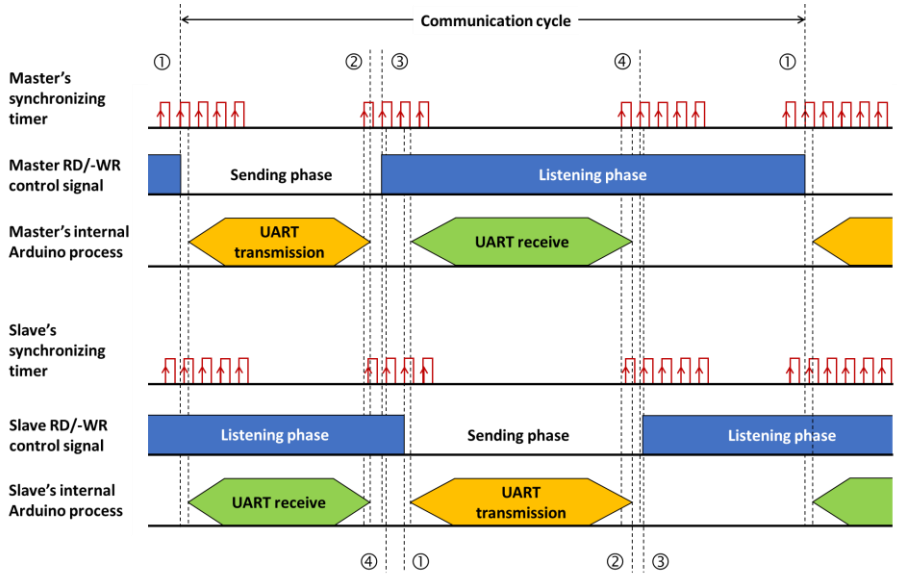

Figure 4. *Data transfer between the Master and a Slave using the proposed algorithm for message transmission and receive. Numbers indicate network events: ①-Start of a transmission; ②-End of a transmission; ③-Communication line release; and ④-Message detection.*

Several algorithms for message transmission and receive have been built so far for Arduino boards [9]-[11]. However, these algorithms mostly run in the main thread program. The incoming message thus cannot be detected quickly, increasing the dead time before response. Consequently, the rate of message sent over the network is limited, suppressing the use of these algorithms for real-time control systems. In this study, the message transmitting and receiving process is separated from the main process and is

carried out regularly to catch the time right after a transmission is completed. In the case of the sender, the Read/Write control signal is then switched from Write to Read almost immediately to release the communication line for other devices. In the case of the receiver, the response message waiting to be sent can be transmitted out as soon as possible. Therefore, the dead time between two consecutive message transmissions is minimized, and the communication cycle can be small enough for high rates of message requested by real-time control systems.

In this algorithm, a high speed timer available in Arduino board is used as a synchronizing trigger source to regularly jump from the current program process to its interrupt service routine where the message transmitting and receiving process runs. In this process, if the Arduino board is in the Listening mode, the serial read buffer is checked for new received bytes, and the pre-defined SOF and EOF characters are searched to detect a complete message. In the Sending mode, the communication line is released if the UART transmission is finished. Generally, there are four network events sequentially occuring as illustrated in Figure 4:

1. *Start of a transmission*: the Read/Write control signal is set to the Write value and the outgoing message is copied to the empty serial write buffer in the message transmitting and receiving process, followed by an UART transmission carried out in the internal Arduino process;

2. *End of transmission*: the UART transmission is finished;

3. *Communication line release*: the Read/Write control signal is set to the Read value in the message transmitting and receiving process;

4. *Message detected*: the message transmitting and receiving process successfully detected a complete message in the serial read buffer, then the destination ID number is checked whether the payload content is safely neglected or not.

Depending on the baudrate of the network selected in a range from several kbps to 1 Mbps, the synchronizing trigger interval can be set correspondingly from 1 ms to 0.01 ms for the fastest response.

4. Results and Discussion

4.1. Automotive application: A supermaneuverable electric vehicle model

The RS-485 Master-Slave network designed above was applied to a miniature model of 4x4 electric vehicle with all-wheel drive and all-wheel steering capabilities as shown in Figure 5. In this vehicle model, an encoder-integrated DC motor is installed at each wheel for independent wheel speed control. Besides, a servomotor-driven steering mechanism is assembled at each wheel for independent wheel steering angle control. As a result of this design, supermaneuverability can be achieved with four steering modes, including front-wheel steering as normal vehicle, all-wheel steering for reduced steering radius at low speeds, diagonal driving for changing lane with zero yaw motion, and zero turn. It is important to be noted that the rotational speed and steering angle of a wheel must be synchronized to those of others to satisfy the steering kinematics required by each steering mode.

Using only one expensive, high performance multi-core microprocessor board that can simultaneously handle all quadrature encoder readings, closed-loop wheel speed controls and wheel steering angle controls is a programming burden to achieve real-time operation. Electrical wiring is also a burden with a lots of wire bunches connecting the microprocessor board to all motor drives, motors, and encoders. By applying a RS-485 Master-Slave network using low-cost Arduino Micro Pro boards, efforts for real-time programming was significantly reduced since all devices could be programmed independently. Electrical wiring was also much simpler because only one pair of wire was used to connect all devices. For each steering mode, the vehicle motion controller (referred to as the Master) calculates wheel's speeds and steering angles needed to ensure that all wheels roll around the same turning center. Then, all calculated speed and steering angle considered as desired setpoint values for closed-loop controls are transmitted via the RS-485 bus wire to all wheel motion controllers (referred to as the Slaves). In each Slave, the speed and steering angle setpoints assigned to its wheel are extracted from the received message frame for further processings. Once the Master's transmission is completed, the Slave 4 immediately response its current actual wheel speed and PWM outputs for motor controls, followed by the response of the Slave 3, then the Slave 2, and finally the Slave 1. Once the Slave 1's transmission is finished, the network comes to a break time and waits for the next Master's transmission.

The time duration between two consecutive Master's transmissions is defined as a communication cycle. This network access protocol is featured by the Master's request, followed by the sequential responses from the Slave that has the highest number to the Slave 1.

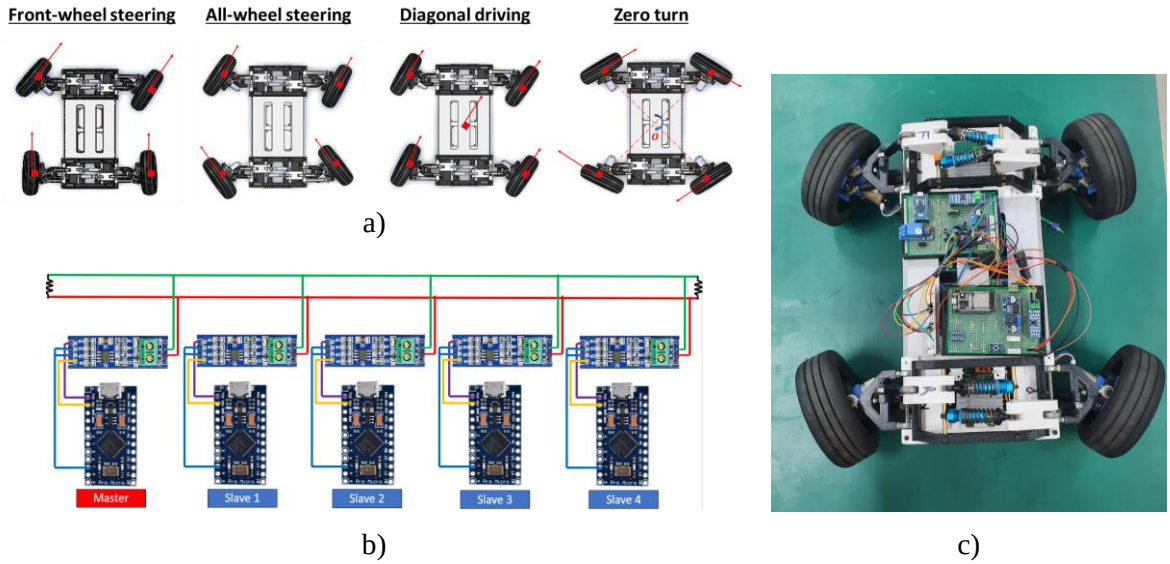

Figure 5. *A miniature 4x4 electric vehicle model with all-wheel drive and all-wheel steering capabilities. a) A illustration of supermaneuverability with four steering modes. b) The RS-485 Master-Slave network designed for the model consisting of five UART/RS-485 interfacing modules, five Arduino Micro Pro boards (one vehicle motion controller (the Master) and four wheel motion controllers (the Slaves)). c) A photo of the actual vehicle model in zero turn mode.*

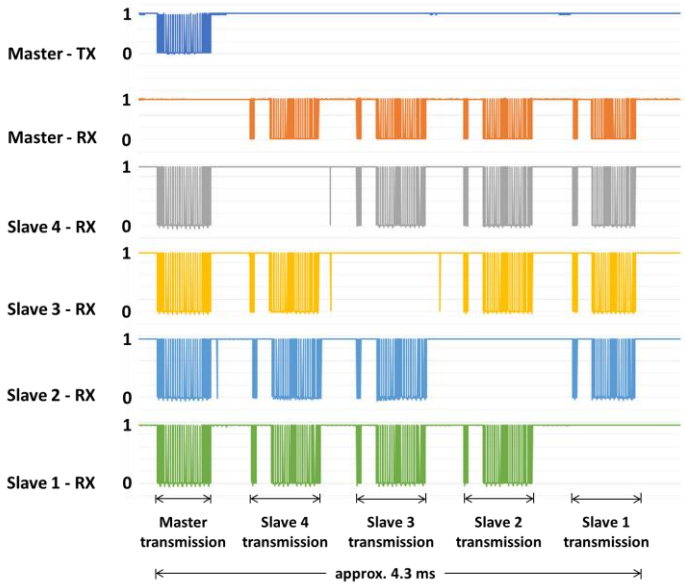

Figure 6. *Signal waveforms measured at the TX and RX pins of the Master and the Slaves during operation. The Master transmission starts a communication cycle and then all Slaves response sequentially. During a communication cycle, data bits are transmitted through the TX pin of the source and received through the RX pins of all the others.*

In this application, the payload length of 16 bytes, twice the payload of a standard CAN protocol. The UART's baudrate was set to 500 kbps, and the checking interval for message transmission and receive control was configurated at 0.2 ms. As a result, the transmission time of a message was approximately 0.6 ms, as shown in Figure 6. With only 5 devices in the network, the time between the start of the Master's transmission and the end of the Slave 1's transmission was approximately 4.3 ms, allowing at least 50 communication cycles per second to achieve smooth vehicle motions. Moreover,

control algorithms in the Master and the Slaves could be freely upgraded or altered. This automotive application revealed that the RS-485 protocol newly developed for Arduino boards could work well with very fast response and customized payloads.

4.2. Environmental application: An IoT-based saltwater intrusion early warning system at river entry

The situation with saltwater intrusion in the main rivers of Ben Tre province and surrounding areas is very complicated, with rapid, sudden, and very deep saline intrusion. According to data from Ben Tre province's Hydrometeorological Center, salinity 4‰ has penetrated 60 kilometers from the river entry. Salinity is greater than 2‰ on average in most districts and cities on tributary rivers, inland fields, and dams temporarily storing water. The salinity suddenly increased and penetrated deeply into the field, causing extensive damage to the rice, crops, and fruit trees. High salinity water has the potential to infect fruit trees with high economic value (such as durian), affecting tree growth and even killing trees if salinity penetrates deeply and for an extended period of time. In the face of the aforementioned situation, effective and long-term solutions to deal with saltwater intrusion and minimize damage to agricultural production are require.

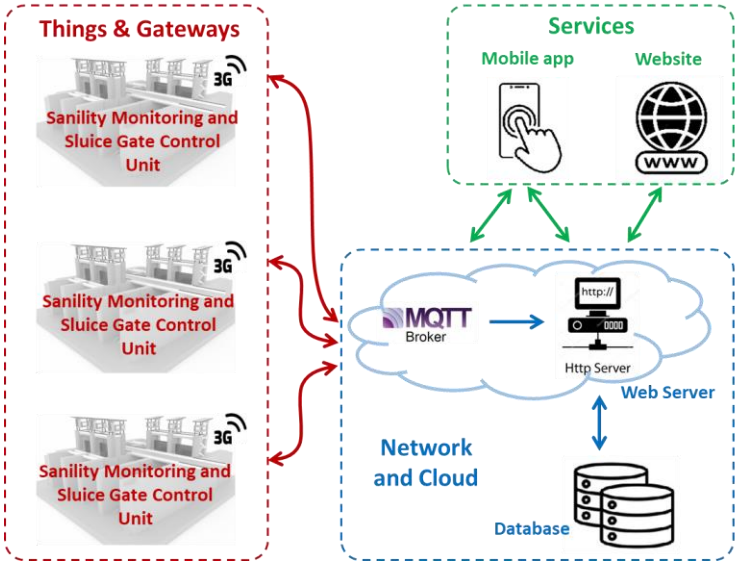

Figure 7. *An IoT-based saltwater intrusion early warning system for Ben Tre province. The salinities at many saltwater intrusion preventing sluice gate areas are continuously measured and stored in a webserver where users can access these data and make appropriate counteractions to prevent saltwater intrusion. Sluice gates can be controlled manually via mobile app or automatically basing on pre-defined salinity thresholds.*

One of the most important solutions is continuous salinity monitoring, which provides saltwater intrusion early warning when salinity exceeds the allowable threshold, allowing people, relevant departments, and local authorities to respond quickly. Salinity monitoring systems are currently in place in the Mekong Delta provinces, but they are mostly located outside of large river tributaries and come at a high cost. According to the Ben Tre province Department of Natural Resources and Environment, the total cost for one monitoring station is around 1 billion VND, and the annual cost of operation and maintenance for 20 stations is around 2.5 billion VND because all sensors, electronic circuit boards, and software must be purchased from abroad with high costs. Therefore, it is critical to design and manufacture salinity monitoring systems in Vietnam that can be operated via an IoT platform and can automatically control sluice gates opening to prevent saltwater intrusion when salinity exceeds certain thresholds (Figure 7). By applying the developed RS-485 protocol to each salinity monitoring and sluice gate control unit (Figure 8), each task of the salinity monitoring and sluice gate control unit can be designed independently, as well as can be further upgraded freely. Moreover, electrical wiring is simple, and noise immunity is significantly improved compared to non-networked solutions. As a result, it is possible to built saltwater intrusion early warning systems with low investment as well as repair costs, and the system can be customized freely.

This environmental project demonstrates that the newly developed RS-485 protocol for popular Arduino boards can be used to implement complex applications at a low cost.

JTE HCMUTE
ISSN: 2615-9740
JOURNAL OF TECHNICAL EDUCATION SCIENCE
Ho Chi Minh City University of Technology and Education
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn

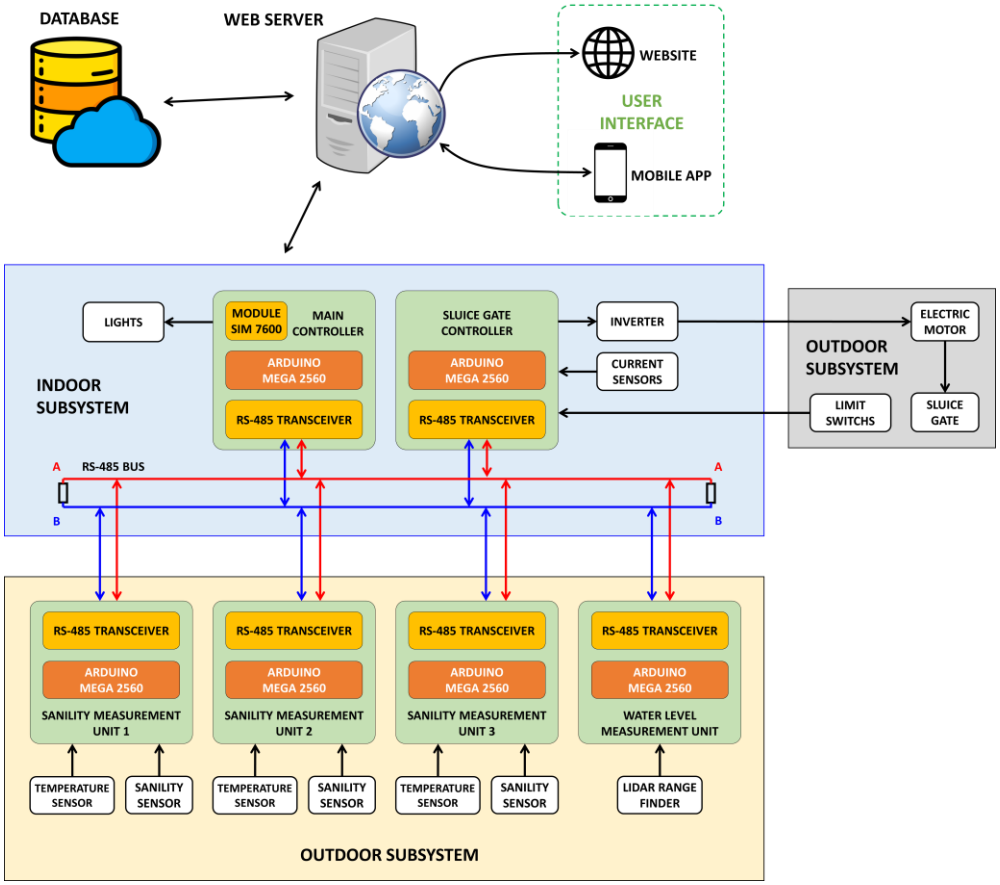

Figure 8. *The schematic diagram of a salinity monitoring and sluice gate control unit at each saltwater preventing sluice gate area. The sluice gate is driven by an AC motor and its opening is checked via limit switches. Temperature-compensated salinity is sampled at three different positions for reliability. Water level and motor currents are also recorded. Each Arduino Mega 2560 boards is used for a specific task such as data acquisition, control, as well as communication to the webserver. Data exchange among Arduino boards is carried out over long distances via the RS-485 protocol newly developed in this study.*

4.3. Discussion

The high response speed and the feasibility of the developed Arduino-based Master-Slave network for complex applications are demonstrated in the previous parts. In this part, network performance is further discussed as follows.

The proposed RS-485 network for Arduino family is mostly based on industrial RS-485 design, RS-485 transceivers, Arduino high speed timer as well as Arduino Serial Library, of which high reliability and stability are well-confirmed. It is therefore safely to induce that network operations are also highly reliable and stable. Moreover, up to 32 devices can be connected to a network and each device can have a different payload length, providing high scalability and flexibility.

Depending on the application, several technical issues must be considered, bringing challenges to designers:

- If more than one Slave transmits at the same time due to incorrect Slave ID assignment, message collisions can occur and go undetected. The network configuration process must be done with caution.
- The maximum payload length is limited to 56 bytes because the lengths of the serial write buffer and the serial read buffer are typically pre-defined as 64 bytes in the Arduino Serial Library. For most of applications, 56 bytes of payload are sufficient to carry a large amount of information instantly exchanging between the Master and the Slave. For heavier payloads, information must be divided into small packages that the Slave sends one by one to the Master. Different network access procedures must be programmed for different Slaves and all must be included in the Master. Programming efforts become significantly high.

- Some Slaves may unintentionally disconnect from the network.
- Additional algorithms should be developed to make the Arduino-based RS-485 network more effective, such as a universal network access procedure that is independent of the number of packages required to be sent by a Slave due to too heavy payload, and a watchdog timer in the Master to detect defected Slaves and trigger alarms.

5. Conclusions

This study demonstrated that real-time data acquisition and control over Master-Slave networks can be realized with a low cost by using Arduino microprocessor boards which can be programmed for RS-485 networking functionality. Mostly based on industrial RS-485 network design, common RS-485 transceivers, Arduino high speed timer as well as Arduino Serial Library, network operations appeared to be highly reliable and stable. In conjunction with fast response and customized payload, the proposed RS-485 protocol significantly extend the usage of popular Arduino boards for complex applications.

In practical applications, designers must be aware of incorrect device IDs and payload limitation. A network access procedure for all payloads and an algorithm to detect disconnected Slaves should be developed to make the network more effective.

Acknowledgments

This research is funded by Department of Science and Technology in Ben Tre province under grant number 1642/HĐ-SKHCN. We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for this study.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

NOMENCLATURES

AWD	All-Wheel Drive
AWS	All-Wheel Steering
CAN	Control Area Network
DL	Data Length
EOF	End of Frame
FSM	Finite State Machine
I^2C	Inter-Integrated Circuit
IoT	Internet of Things
PWM	Pulse Width Modulation
RX	Receive
SOF	Start of Frame
SPI	Serial Peripheral Interface
TX	Transmit
UART	Universal Asynchronous Receiver/Transmitter

REFERENCES

[1] A. Holovatyy *et al.*, “Development of Arduino-Based Embedded System for Detection of Toxic Gases in Air,” *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 2018, pp. 139-142, doi: 10.1109/STC-CSIT.2018.8526672.

[2] A. Macker, A. K. Shukla, S. Dey, and J. Agarwal, "ARDUINO Based LPG Gas Monitoring … Automatic Cylinder Booking with Alert System," *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, 2018, pp. 1209-1212, doi: 10.1109/ICOEI.2018.8553840.

[3] M. El-Abd, “A Review of Embedded Systems Education in the Arduino Age: Lessons Learned and Future Directions,” *International Journal of Engineering Pedagogy*, vol. 7, no. 2, pp. 79-93, Apr. 2017, doi: 10.3991/ijep.v7i2.6845.

[4] A. Gonzálezn *et al.*, “A low-cost data acquisition system for automobile dynamics applications,” *Sensors*, vol. 18, no. 2, pp. 366-386, Jan. 2018, doi: 10.3390/s18020366.

[5] T. H. M. Nguyen *et al.*, “Design of wifi-based remote monitoring and control system,” *Vietnam Journal of Science and Technology*, vol. 62, no. 11, pp. 45-48, Nov. 2020.

[6] A. Ma'arif, “Control of dc motor using integral state feedback and comparison with PID: Simulation and Arduino implementation,” *Journal of Robotics and Control*, vol. 2, no. 5, pp. 456-461, Sep. 2021, doi: 10.18196/jrc.25122.

[7] M. Tastan, “A low-cost air quality monitoring system based on Internet of Things for smart homes,” *Journal of Ambient Intelligence and Smart Environments*, vol. 14, no. 5, pp. 351–374, Jul. 2022, doi:10.3233/AIS-210458.

[8] RS-485 serial interface explained, Jason Kelly, [Online] Available: https://www.cuidevices.com/blog/rs-485-serial-interface-explained.

[9] V. Tipsuwanporn *et al.*, "Development of redundant bus library for arduino to apply in SCADA system," *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, Gyeonggi-do, Korea (South), 2014, pp. 42-46, doi: 10.1109/ICCAS.2014.6987955.
[10] Sending Data Between Two MKR 485 Shields, Karl Söderby, [Online] Available: https://docs.arduino.cc/tutorials/mkr-485-shield/mkr-485-communication/.
[11] Modbus (RS-485) Using Arduino, Houma-Hackster, [Online] Available: https://www.hackster.io/maurizfa-13216008-arthur-jogy-13216037-agha-maretha-13216095/modbus-rs-485-using-arduino-c055b5.

Tran Dang Long received the Bachelor's degree in Automatic Control and the Master's degree in Technology Cybernetics from Ho Chi Minh University of Technology (HCMUT) in Vietnam, in 2002 and 2005, respectively. In 2017, he was awarded a PhD. degree in Hydrogen Energy Systems from Kyushu University in Japan.
From 2002 to 2023, he serves as a researcher and academic in the Faculty of Transportation Engineering, HCMUT. His research interests include engine control, ADAS, electric vehicles, and fuel cell technologies.
Email address: trandanglong@hcmut.edu.vn. ORCID: https://orcid.org/0000-0002-5839-3148

Nguyen Truong Hoa Binh just recently graduated as a Bachelor of Automotive Engineering from Ho Chi Minh University of Technology (HCMUT) in Vietnam in 2023. He was awarded with the Talented Student in 2020 and was an attendee of various volunteer activities like Green Summer and Voluntary Spring Campaign in Vietnam. During his studying, he was experienced as a lab-assistant in the Bosch Automotive Lab of HCMUT. Email address: nthoabinh6501@gmail.com. ORCID: https://orcid.org/0009-0001-1066-8368

Ho Nam Hoa earned his Bachelor's degree in Automotive Engineering from Ho Chi Minh University of Technology (HCMUT) in Vietnam in 2020. Currently, he is a Master student of Vehicle Engineering in HCMUT and a researcher in the Faculty of Transportation Engineering in HCMUT. His research interests encompass embedded control systems, intelligent control, and mechatronics. Email address: namhoaho@hcmut.edu.vn. ORCID: https://orcid.org/0009-0005-1395-430X

Nguyen Duy Anh is a researcher and academic in the fields of Mechatronics engineering, automation, robotics and logistics. He obtained his Bachelor's degree in Automatic Control from Ho Chi Minh City University of Technology (HCMUT), Vietnam in 2003. He furthered his studies by pursuing a Master's degree in Logistics from Korea Maritime University, completing it in 2006. In 2009, he was awarded a PhD. degree, also in Logistics, from Korea Maritime University. Currently, he serves as an Associate Professor at the Faculty of Mechanical Engineering, Ho Chi Minh City University of Technology (HCMUT) in Vietnam. His research interests encompass logistics, automation, robotics, mechatronics, computer vision, and manufacturing technologies. Email address: duyanhnguyen@hcmut.edu.vn. ORCID: https://orcid.org/0000-0002-5280-8453

Tran Van Danh received a Bachelor's degree in Control and Automation Engineering at Ho Chi Minh City University of Technology and Education in 2019. Currently, he is working as a Researcher for Institute for Nanotechnology (INT), Vietnam National University – Ho Chi Minh City (VNU-HCM). His research interests include Embedded Systems (Programming & PCB Layout), Real Time Operation System (RTOS), and Internet of Things (IoT) for real-life applications. Email address: tvdanh@vnuhcm.edu.vn. ORCID: https://orcid.org/0000-0002-5233-8025

Nguyen Minh Nhat obtained his Bachelor of Engineering degree in Control Engineering and Automation from Ho Chi Minh City University of Technology, Viet Nam National University Ho Chi Minh City, VietNam in 2022. Currently He is a engineer with Institute for Nanotechnology, Vietnam National University Ho Chi Minh City in Vietnam. His work has focused upon designing schematic circuit for hardware circuit and 3D models, programming software for micro-controllers and applications. Email address: nmnhat.int@vnuhcm.edu.vn. ORCID: https://orcid.org/0000-0001-8072-6677

Doan Duc Chanh Tin obtained his Master degree in Materials Technology in 2005 from Ho Chi Minh City University of Technology, Vietnam National University Ho Chi Minh City, Vietnam. He obtained his Ph.D. diploma from Wageningen University, the Netherlands in 2012. His Ph.D. research focused on development of polymer-based sensors for carbon dioxide detection in greenhouses. Currently, he is Director of Institute for Nanotechnology, Vietnam National University Ho Chi Minh City, Vietnam. He is author and co-author of 1 thesis book, 47 articles in peer-reviewed international journals, 07 articles in national journals, 66 proceedings papers of international conferences, 10 patents and 5 applied patent applications. His research interests include micro-nano sensors for environmental and biological applications, materials and microsystems for energy applications. Email address: ddctin@vnuhcm.edu.vn. ORCID: https://orcid.org/0000-0003-4250-9078