Design of Communication Protocol based on 9 Bit UART on ADROIT Education Robot

Eko Henfri Binufroho[1], Endah Suryawati Ningrum[1], Adnan Rachmad Anom Besari[2]
[1]Mechanical and Energy Engineering Department
[2]Computer and Informatics Engineering Department
Electronic Engineering Politechnics Institute of Surabaya
{sragen, endah, anom}@pens.ac.id

Abstract

As modularity became essential in educational robot, communication to be an important part in connecting electronics modules in the robot itself. To reduce overall cost, a low-cost microcontrollers have been used as the controller on each of the robot modules. Most of low-cost microcontrollers have been equipped with UART as the communication interface. But UART interface standard only describes the physical layer of the communication. The design of the higher level communication layer will be up to the user implementation. Beside that, UART interface basically just designed for poin-to-point communication. So for multidrop communication, a special communication protocol is needed. This paper proposes a simple and easy to implement full-duplex data link and network layer protocol stacks based on single-master and multi-slaves configuration on multidrop communication bus. Using this design, up-to 64 robot modules can be deployed on the proposed communication line.

Keywords: Education robot, communication protocol, UART, data-link layer.

1. Introduction

Robotics has been used widely in education as a learning tool, as it provides a motivating and interesting tool to perform laboratory experiments within the context of mechatronics, electronics, microcomputer, and control [1]. Many studies have been conducted in the investigation in the field of educational robotics and identification of the new challenges and trends focusing on the use of robotic technologies as a tool that will support creativity and other learning skills this last decade [2]. Educational robotics is an educational activity that support and strengthen specific areas of knowledge and skills developed in students through the design, creation, assembly and operation of robots. The goal of teaching robotics also to teach student to adapt the current trends in automation technology which is related to the use of mechanical, electronic and computer-based, in the operation and control of the production plays a very important role.

In robotics education, modularity becomes an important part. Modularity is the degree to which a robot system's components may be separated and recombined. It gives student flexibility in combining and creating a robotics design to explore the student creativity within the given project. This modularity makes the student understand the function and working principle of each part more easily. The greater the degree of modularity makes the greater of flexibility. But it also makes the system more complex. Such a new challenge that arises is how to connect each of electronic modules together. Protocol becomes an important component to control the data transfer during the communication.

Communications and its protocols will play an important role in mobile robot systems to be able to address the real world applications. A poorly implementation of communications protocols, or protocols which are poorly matched to the functional and performance characteristics of the underlying physical communications links, can greatly reduce the effectiveness of an otherwise well implemented robotic or networked sensor system. This paper proposes a design of communication interface for ADROIT V1, an educational robot kit that designed in a modular form. Each electronics module has its own controller which communicates using a multidrop bus with single-master and multiple-slaves configuration. The communication signal is in a TTL level based on UART Interface. Combined with the designed communication protocol, this design shows a simple but efficient communication between the main controller and other modules in ADROIT V1.

2. Related Works

Universal Asynchronous Receiver and Transmitter (UART) is an integrated circuit which plays an important

role in serial communication. The UART is a standard communication component that is provided by most of the available microprocessors and microcontroller [3]. UART has been implemented in many robotics applications. It has been used in communication of modular join control system of humanoid robot which combines the capability of high-speed data processing of DSP TMS320F2812 for coordination layer and the predominant control performance of 8-bit MCU for execution layer [4].

UART-based communication also has been used in robotics commercial products such as Dynamixel servo motor from Robotis. Dynamixel motors have been used in many robotic projects [5-7]. These motor doesn't use PWM signal for its position control. It uses a half-duplex, one-wire, RS-232 protocol-based TTL communications link transfers command and status information between a host controller and the robot actuators. Since the communication is in half-duplex, it means that devices attached to a common communications link are only allowed to talk when no other device is talking. In the case of the Dynamixel robot actuators, all of the robot actuators are daisy-chained on the one-wire link and spend most of their time listening and only speak after being spoken to [8].

3. The Communication Design

The overall ADROIT V1 educational robotics module can be seen from Figure 1.

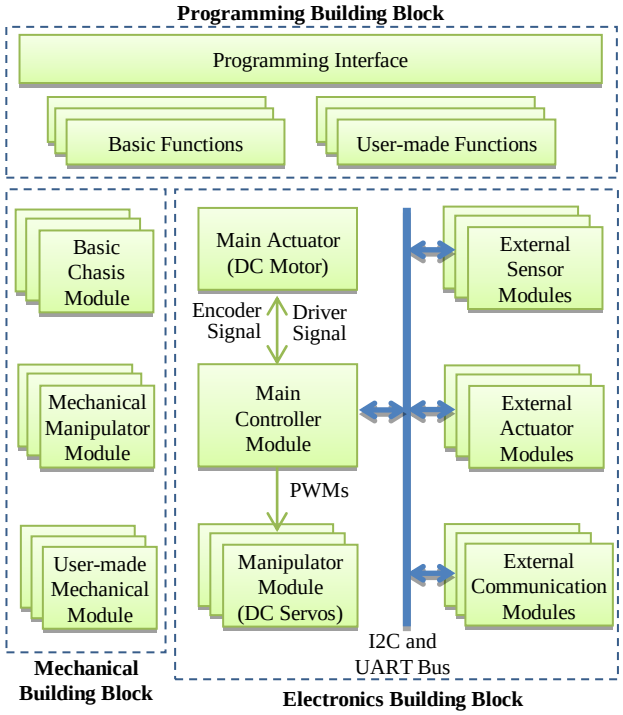

Figure 1. ADROIT V1 Building Blocks

ADROIT V1 educational robotics module composed from as three main parts; mechanical building block, electronics building block and programming building block. The proposed communication become one of the main communication buses used to connect the main controller to external sensor, actuator and communication modules inside the electronics building block. The design of this communication protocol stack consists of physical data link layers, and network layer. This protocol will control the packet data transfer between master and slaves in the communication bus.

3.1. Physical Layer

The proposed UART communication parameter is 9 Bit length, 1 Stop bit, and using Even Parity. The UART baudrate is 38400 bps. The selection of this baudrate is based on the trade off between speed and stability of the overall tested system. The higher the baudrate the higher the speed, but the stability could become lower. The communication signal is TTL level, which is defined as "low" when the voltage is between 0 V and 0.8V with respect to the ground terminal, and "high" when between 2.2V and 5V. The TTL level is used to make the hardware implementation become very simple, since this voltage can be driven directly from the microcontroller without additional line driver.

The communication line is designed to support multi-drop bus, in which all components are connected to the same electrical circuit. This method is generally used when many devices need to be connected to a single bus. The proposed design uses a single-master and multiple-slaves configuration. It supports Full duplex serial communication protocol, where both TxD and RxD can be used at the same time. The full-duplex only available between master and one active slave in a given time. The hardware implementation to support this method is very simple. Master's TxD is connected to all slave's RxD, and master's RxD connected to each slave's TxD through the reversed diode as depicted in Figure 2.

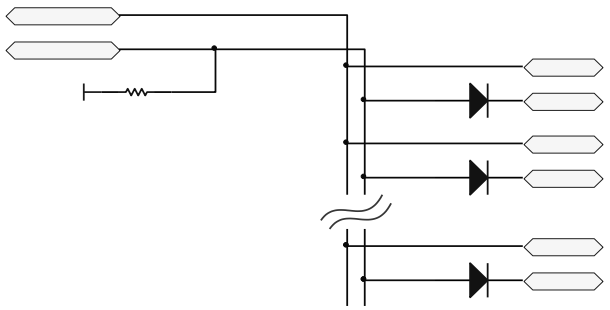

Figure 2. Hardware connection

The use of 9 bit communication in the design refers to RS-485 protocols. This additional bit made process to distinguish data byte and address byte more easily. Data

packet from the master consists of address byte within the transmitted frame that identifies the target slave device. With early-generation microcontrollers, all slaves had to search for this address byte within the frame to determine whether the frame was addressed to them. This meant that every slave in the network had to be interrupted by every byte being transmitted, wasting the slaves' precious CPU processing time. But now, most of modern microcontrollers, like AVR and PIC implemented their serial port with a special mode for multiprocessor communications. In this mode, 9 bits are received. The 9th bit goes to a special register. The serial port can be configured to activate the serial-port interrupt, depending on the state of the 9th bit.

Each time, the master communicates with only one of the slaves. When the master wants to transfer a block of data to a slave, it first sends out an address byte to identify the target slave. The 9th-bit of the data byte sent from the master is set to 1 to indicate the address byte while cleared to 0 to indicate the data byte. All the slave systems will compare the address byte with their own address. Only the target slave will respond to the master. The master then starts transmitting data bytes to the target slave. The non-addressed slaves will continue with their program, ignoring the incoming data bytes until a new address byte interrupts them, minimizing wasted its CPU processing time.

3.2. Packet Format

In the proposed packet is designed as a light as possible. The number of bytes in each packet is minimized to reduce the protocol overhead, ranging from only 2 bytes up to maximum 299 bytes. Each frame also contains the network layer in the protocol stack. The designed communication protocol embeds the Packet ID or Response ID to reduce the number of packet bytes transferred. This method is useful since Short Packet and Short Response data transfer are frequently used in ADROIT. The data packets are divided in to two section, packet from master controller as Command Packet, and packet from the slave controller as Response Packet. The structure of the Command Packet be seen from Table 1.

Table 1. Structure of the Command Packet

Byte Offset	Value	Description
#1	Address = 0 - 63 (bit 0 - 5) Bit 6,7 = Packet ID	Slave Address and Packet ID
#2	0-0xFF	Command ID
#3	n = 1 - 255	Payload Length
#4 – #(3+n)	0 – 0xFF	Payload
#(4+n)	0 – 0xFF	Checksum (optional)

The detail of the packet is as follows:

- The **Slave Address** is the address of the slave controller who should response the packet. The value is between 0-63 for normal packet. The value of the bit 6 and 7 show the Packet ID which mean:
  - 00 = normal data packet
  - 01 = Short packet without payload
  - 10 = Short packet with 1 byte payload
  - 11 = Broadcast message to all slave controllers
  - In **normal data packet**, master will send the full packet format, including Payload Length, Payload data followed by optional Checksum byte.
  - In **short packet without payload**, the Payload Length and Payload bytes did not exist. This packet is used to reduce frequent packet which mostly used in ADROIT when the controller just want to send a simple command or request to the slave controller.
  - In **short packet with 1 byte payload**, the 3rd byte will indicate the single Payload from the master. This packet is extension from the short packet type that is also frequently used in ADROIT.
  - In **broadcast message**, the massage will be received by all slave controllers, regardless the value of the **Slave Address**. This kind of command could be used to send a reset command to all slave controllers.
- The **Command ID** identifies the command from the master to the specific slave controller. Slaves have a common command code, but some slaves also have their own specific command codes depend on specific task that can be handled by each of them.
- The **Payload Length** identifies the number bytes of the payload, including the checksum if it enabled. This byte is only available in normal packet type.
  In the current protocol, the payload length only reserved up to 255 byte only.
- The **Payload** contains meaningful values that correspond to the particular Command ID. This byte(s) is available in normal packet type.
- The **Checksum** contains the byte of sum of all bytes starting from the command ID to the last payload byte. This checksum can be enabled or disabled, and automatically disabled in the short packet mode.

The structure of the Response Packet has some differences from the Command Packet. The detail configuration can be seen from the table 2.

Table 2. Structure of the Response Packet

Byte Offset	Value	Description
#1	Address = 0 - 63 (bit 0 - 5) Bit 6,7 = Response ID	Acknowledgement
#2	n = 0 - 255	Payload Length
#3 – #(2+n)	0 - 0xFF	Payload
#(3+n)	0 - 0xFF	Checksum (optional)
#(4+n)	0xF0 – 0xFF	End of Packet

The detail of the packet is as follows:

- The **Slave Acknowledgement** sends the address of the slave that should response the packet from the master. This byte also becomes the acknowledgement byte indicating the slave has already received the address and recognized the Command ID sent by the master. The value is between 0-63 for normal packet. The value of bit 6 and 7 show the Response ID which mean:
  - 00 = normal packet response
  - 01 = Short response with 1 byte payload
  - 10 = Short response with 2 bytes payload
  - 11 = unrecognized command ID
  - In **normal packet response**, the response will send the full format, including Payload Length, Payload data and optional Checksum byte.
  - In **short packet response**, the Payload Length and Payload bytes did not exist. This packet is used to reduce frequently used packet in ADROIT, such as to read data from simple sensor or actuator. The number of response is only 1 or 2 byte(s) length.
  - The **unrecognized Command ID**, confirm to the master controller that the addressed slave could not recognized the command sent to it, thus master can take other action.
- The **Payload Length** identifies the number response bytes of the payload, including the checksum if it enabled. This byte is available in normal packet response. In the current protocol, the payload length also only reserved up to 255 byte only.
- The **Payload** contains the response bytes correspond to the particular Command ID sent to the slave controller. This byte(s) only available in normal packet response.
- The **Checksum** contains the byte of sum of all bytes starting from the Payload Length byte to the last payload byte. This checksum can be enabled or disabled, and automatically disabled in the short packet response.
- The **End of Packet** contains the byte massage to report the master controller about the Package Transfer information. The message reports the transfer condition, command execution process, checksum result or any other condition that occur, whether it has been succeeded or failed.

3.3. Error Detection and Handling

The delay time between bytes in the packet will trigger the communication timeout. If the delay time is over 100ms, then the controller recognizes this as a communication problem as depicted in Figure 3. Slave controller will automatically send this status to the master controller and it will restart the packet detection from the beginning. The same process also occur in the master, when the delay of the inter-byte response exceed 100ms, it will restart the Command Packet. The the same error handling mechanism will be triggered when the parity error is detected.

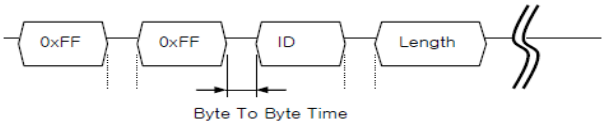

Figure 3. Delay between bytes

When the checksum is enabled, slave will discard the data if the checksum error is detected, and it will send the error report to master controller using the End of Packet byte. Master will retransmit the packet if necessary.

3.4. Packet Decoding

Slave controller will detect the address and the Packet ID before proceeds the response. The decoding process in the programming uses a look up table method to make the execution faster. The algorithm used by the slave controller is shown in the flowchart in Figure 4.

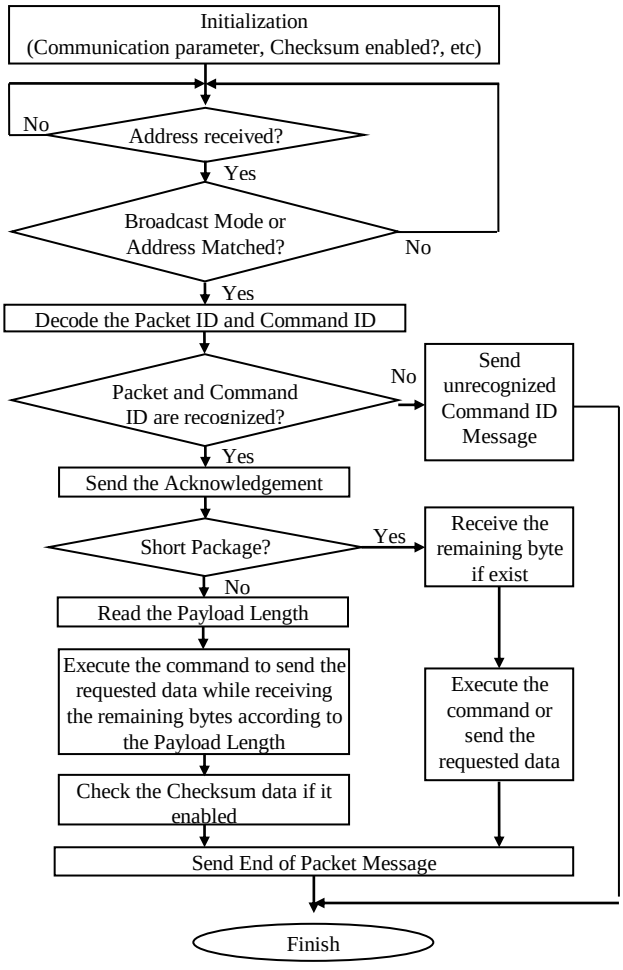

Figure 4. Decoding and execution algorithm used in the slave controller.

4. Experiment Result

Performance of the proposed communication protocol is indicated by measurement of the communication latency and information throughput. The data setup latency in the slave controller is minimized by directly accessing the data within slave's memory buffer.

Table 3. Communication latency

Packet Type	Number of payload (bytes)	Ack. Latency (ms)	Command Packet Transfer Time (ms)
Normal Packet	5	0,9	3,04
	10	0,9	4,74
	50	0,9	18,28
	100	0,9	35,2
	200	0,9	65,06
Short Packet with no payload	0	0,9	1,23
Short Packet with 1 byte payload	1	0,9	1,577

Table 3 shows the acknowledgement latency which measured since the Command Packet is sent until the acknowledgment is received. In all packet types, the acknowledgement latency has the same result at 0.9ms. The result shows that the slave controller can directly send the acknowledgement after it recognized the Command ID in the 2nd byte of the packet. The total transfer time or packet latency is measured until master has received End of Packet byte from the slave controller.

To measure the information throughput, the slave is made to send same number of payload from its buffer memory. Payloads will be sent in full-duplex, in which master and slave send the payload in almost the same time. The result of this test can be seen from Table 4. The information throughput is calculated based on the total number of payload can be transferred within the packet divided by the round-trip time. Protocol overhead is calculated by dividing additional byte in the frame with the total payload transferred.

Table 4. Information Throughput

Packet Type	Protocol Overhead (%)	Number of payload (bytes)	Round-trip Time (ms)	Information Through-put (bytes/s)
Short Packet with 1 byte payload	100	1	1,24	1612,9
Normal Packet	60	5	3,72	2688,17
	30	10	5,41	3696,85
	6	50	18,95	5277,04
	3	100	35,88	5574,13
	1.5	200	69,74	5735,58

Table 4 shows that the full-duplex mode is very effective in increasing information throughput in Normal Packet data transfer, especially in increasing number of payload. But in Short Packet data transfer, the total response is same with the previous test. It can't take the advantage off full-duplex capability since the number response byte is very small as the response will be sent ahead to the master controller after slave controller received the last payload.

5. Conclusion

This paper has proposed a multi-drop communication protocol designed for ADROIT V1 education robot. The communication handled by each controller in each slave that uses a low-cost microcontroller. The packet format is designed to accommodate simple to more complex data transfer in regards to data characteristic in ADROIT modules. The full-duplex capabilities can give a better response to system in which acknowledgement from the slave controller can be sent in advance before the full packet bytes has been received from the master controller. The full-duplex feature has an advantage in Normal Packet data transfer, in which data can be sent in both directions. The advantage of this feature becomes bigger in the increasing number of payload data as the protocol overhead become lower.

6. Acknowledgement

This research was conducted by EEPIS Robotics Research Center (ER2C) in Electronic Engineering Polytechnic Institute of Surabaya (EEPIS). This work was supported by EEPIS Center for Research and Community Service (*Pusat Penelitian dan Pengabdian kepada Masysrakat, Politeknik Elektronika Negeri Surabaya*), and Ditlitabmas Dikti.

References

[1] Li Xuemei, Xu Gang, "Interdisciplinary Innovative Education Based on Modular Robot Platform", *ISECS International Colloquium on Computing, Communication, Control, and Management*, pp. 66-69, Guangzhou, August 2008.

[2] Alimisis, D., Arlegui, J., Fava, N., Frangou, S., Ionita, S., Menegatti, E., Monfalcon, S., Moro, M., Papanikolaou, K. and Pina, A. "Introducing robotics to teachers and schools: experiences from the TERECoP project", *J. Clayson and I. Kalas (eds.) Proceedings for Constructionism 2010,* 16-20 August, 2010, Paris, France.

[3] Ching-Chang Wong and Yu-Han Lin, "A Reusable UART IP Design and its Application in Mobile

Robots", *Mobile Robots, Moving Intelligence*, pp. 147-156, Germany, December 2006.

[4] Zhang Jing, Xiang Zhongfan, Wang Jinge, Liu Zaixin, and Luo Xudong, "The Modular Design of Joint Control System for Humanoid Robot", *International Conference on Multimedia Technology (ICMT)*, pp. 1-3, Ningbo, 2010.

[5] Quigley, M.,Asbeck, A., Ng, A, "A Low-cost Compliant 7-DOF Robotic Manipulator", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6051 - 6058, Shanghai, May 2011.

[6] Inyong Ha, Yusuke Tamura & Hajime Asama, Development of open platform humanoid robot DARwIn-OP, *Advanced Robotics*, Taylor & Francis and The Robotics Society of Japan, Vol. 27, Issue 3, pp. 223-232, February 2013.

[7] Max Schwarz and Sven Behnke, "Compliant Robot Behavior using Servo Actuator Models identified by Iterative Learning Control", *RoboCup 2013: Robot World Cup XVII*, pp 207-218, 2014.

[8] Fred Eady, "Unwinding The AX-12+ Communication Protocol", *Servo Magazine*, pp 30-37, April 2009.