International Journal of Mechanical Engineering
ISSN : 2277-7059 Volume 1 Issue 1
http://www.ijmejournal.com/ https://sites.google.com/site/journalijme/

Realization of CANopen communication control for underactuated anthromorphic finger

Mohamad Khairi Ishak[1] , Jamaludin Jalani[1]

[1] Mechanical Engineering, University of Bristol, Queen's Building, University Walk, Bristol, BS8 1TR
{memki,meyjj}@bristol.ac.uk

Abstract. This paper proposes an alternative control communication system through CANopen application which will be used for controlling an underactuated anthromorphic fingers. It is anticipated that the CANopen network can be developed easily and reliable to integrate with Bristol Elumotion Robot Hand (BERUL). The real-time network has to incorporate into dSPACE and a well-known Matlab Simulink-based controller prototyping system. Experimental result has proved that the CANopen is reliable to be implemented for underactuated anthromorphic fingers.

Keywords: real-time, control communication system, underactuated anthromorphic finger.

1 Introduction

Research in robotics and control engineering are blending uniquely into a challenging area, the development of embedded networked real-time applications. Hence, the applications should timely deliver synchronized data-sets, minimize latency in their response and meet their performance target in the presence of disturbances.

Research in the robotics area has matured to the point where thousands of robot systems are being proposed. In order to achieve a given task, the robots will share sensor and actuator information, e.g torque, velocity, current and position. In this scenario, the communication between the robots and sub-units is one of the main factors which contribute to the capabilities and effectiveness of robots.

Networked communication is a backbone to accomplish all tasks in embedded robot systems. With that, there are various types of communication protocols such as Controller Area Network (CAN), CANopen, Flexray, Ethernet, Time Triggered, Time Triggered CAN (TTCAN) and Time Triggered Ethernet (TTEthernet). Today, an embedded communication network offers flexibility of the designed system and reduces wiring complexity which leads to use embedded networks for safety critical applications. Most of the applications like robotics, manufacturing, medical, military, and transportation systems, depend on embedded computer systems that interact with the real world. The many fields of application come with different requirements on such embedded systems. Messages in distributed embedded system can be sent at a certain time or when an event occurs, known as time-triggered and event-triggered communications.

In a robotic perspective, HUBO robot was developed by the Korea Advanced Institute of Science and Technology (KAIST) using Controller Area Network (CAN) architecture to distribute information between controllers and sensors [1]. Another humanoid robot is LOLA [2] [3] which is still in the development process at the Technical University of Munich focusing on fast and human like walking motion. An Ethernet-based protocol has been used in the communication architecture for this system. These examples show the dependable reactive systems that have to provide a critical real-time service requiring careful design and implementation.

In the real time communication network, the key characteristic is deterministic processing in the system. There are various communication protocols for such purposes which are used in robotics and in industrial areas. Some of the more representative real-time communication protocols today are TTCAN (Time-Triggered CAN) [4] [5] [6], TTP/C (Time-Triggered Protocol) [7], FlexRay [8], CAN (Controller Area Network) [4], and CANopen [7]. The ideas of Time Triggered Architecture (TTA) were developed by Hermann Kopetz and colleagues at the Technical University of Vienna [9]. This protocol architecture is being applied for safety critical applications especially in the aircraft industry.

Particularly, we have developed a design control communication based on CANopen network. BERUL is used to test and evaluate our communication structure to move the fingers.

2 Control Communication Platform

A. CANopen

The CANopen protocol was developed within CAN-in-Automation (CiA) and it is a higher layer protocol for CAN based networks [10]. A first version of the CANopen communications profile, CiA DS-301, was released in 1995.

1) Object dictionary: The main element in the CANopen device is the object dictionary. It is essentially a grouping of objects stored in a lookup table. It can be accessed from the network through a 16-bit index and an 8-bit subindex for individual data structure elements.

The CANopen specifications define several device profiles that describe all the communication parameters and object dictionary entries supported by a specific type of CANopen module. A description of an object dictionary entry typically as below [7]:

- Index: The object dictionary index.
- Object: The object type (Variable, Array, etc.).
- Name: The name of the entry.
- Type: The data type (Integer16, Boolean, etc).
- Access Attributes: Read and write attributes.
- Category: Indicates if the entry is mandatory (must be implemented), optional (may be implemented) or conditional (must be implemented if certain other entries or features are implemented).

2) Service Data Objects: SDO is one of the principal objects in CANopen which allows to access the data stored in a node's object dictionary. Besides that, SDOs also can be applied to write the configuration of data to a node's object dictionary, (e.g. setting device parameters, assign application data to PDOs and assign message identifiers). This allows for a configuration tool to adapt systems behavior dynamically. SDO communication follows the client/server model where every node implements a server to handle read/write requests to its object dictionary. The node that requests data is called the client [10].

3) Process Data Object: PDO is used to transfer real time data which allows transferring 8 data bytes. There are two types of PDO, Transmit PDO (TxPDO) and Receive PDO (RxPDO). TxPDO and RxPDO are used for transmitting data from and receiving data to a device object dictionary. Their functionality and contents are described by a number of parameters stored at predefined indices in the object dictionary [10].

4) Mapping parameters: PDO mapping parameters are stored in the communication profile area in the object dictionary of a client or server. The data content of each PDO is described by a mapping parameter. A PDO mapping parameter specifies the contents of a PDO by the index and subindex of the communicated object dictionary entries. The maximum amount of data transferred by a PDO is 8 bytes where each bit of the PDO mapping can be set individually.

B. Transmission

PDO transmissions conform to the producer/consumer relationship. A TPDO is transmitted to the network by the PDO producer and received as a RPDO by zero or more PDO consumers. The PDO write service uses the push model where the producer performs an unconfirmed data transmission which is received by zero or more consumers; the data is pushed from the producer to the consumer(s). A PDO read is initialized by a consumer sending a Remote Transmission Request (RTR) frame to the network. When the producer of the requested

PDO acquires the RTR, it transmits the PDO and all consumers of the PDO receive it. This approach is called the pull model since the consumer pulls data from the producer. The transmit trigger method decides when a PDO is sent to the network. In CANopen four different methods can be distinguished [10]:

- *Event driven :*

The transmission of a PDO is triggered by the occurrence of an event. The definition of an “event” is usually defined by the device profile. A typical event is a change in the process data.

- *Timer driven :*

If an event has not occurred within a specified time frame, an event is triggered by a timer and a PDO is transmitted.

- *Synchronized:*

The synchronized polling mode is intended for applications where it is crucial to get all inputs at the same time and apply all outputs at the same time. To achieve this, a SYNC signal is transmitted, usually by the network management master, on a fixed time basis. All synchronized nodes keeps their transmit buffer updated with the current data. Reception of a SYNC message triggers transmission of the data.

C. Network Management

Network management (NMT) in CANopen follows the master/slave relationship. One device in the network acts as NMT master controlling the state of the slave nodes with NMT master messages. To realize the use of different operating states, every NMT slave has to implement a state machine. The three major states of NMT slaves are:

Stopped: No communication is allowed in this state except guarding and heartbeat

Preoperational: In this state SDO communication is possible and configuration of the node may be done. PDO communication is not allowed. A node automatically enters this state after initialization.

Operational: In this state all communication is active. All PDOs are created according to mappings in the object dictionary. When a node is started, CAN/CANopen interfaces are initialized and communication parameters are setup. When the initialization phase is completed the node sends a boot-up message and transits to preoperational mode. The boot-up message tells the NMT master that the node is initialized and in preoperational mode. When a node is initialized and in preoperational state it is the NMT masters task to control all state transitions [10].

D. Hardware

The dSPACE is a real time hardware platform which integrates with Matlab /Simulink software allows the real time tests. It has been developed by dSPACE GmbH. By using dSPACE, it allows the user to run the model (controller) which has been designed using Simulink in real time. With the Simulink environment, it is flexible software that includes a large library of control blocks in block graphic language. Apart from that, it also allows for the creation of custom logic in more advanced in designed model and complex to model in Simulink blocks, known as S-function. The test model can be downloaded the generated real-time application in the environment ControlDesk, from dSPACE which provides several instruments for user to input or display data. Also, ControlDesk can read and write the variables of the real-time application.

In order to examine the proposed CANopen protocol, the Bristol Elumotion Robot Hand (BERUL) (see Figure 1) is used. The hand consists of 9 degree of freedom and lightweight 740g. All the hand actuators are situated within its volume, eliminating any transmission problems when the hand is mounted on an articulated wrist. The hand is actuated using MAXON motors fitted with precision gearing with encoders enabling high accuracy positioning.

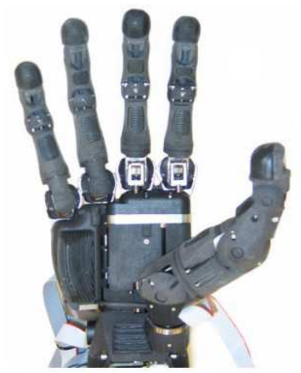

Fig 1. Bristol Elumotion Robot Hand (BERUL)

In communication aspect, DS4302 CAN interface board is used to communicate between the host PC and dSPACE hardware. This board allows to connect the dSPACE system to the CAN bus. In the dSPACE Simulink environment, RTI CAN blockset is available to use in the simulink model. All the CAN interfaces in Simulink can be configured using a dialog-based method, for example to configure a time interrupt. Apart from that, dSPACE-DS1006 Processor Board includes with serial interface RS232 interface with standard UART allowing transfer rates of up to 115.2 kBaud.

3 Implementing CANopen protocol

The aim of this task is to apply the CANopen protocol in order to test the control communication system for MAXON motor based on velocity mode on the BERUL. Figure 2 shows the flow to configure the velocity mode on the finger unit.

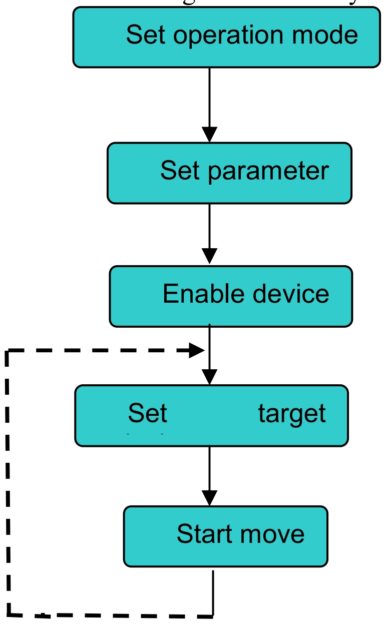

Fig 2. Stateflow of velocity mode profile

The simulink setup enables the CANopen protocol to be used with Embedded MATLAB Function blocks. In the setup simulink blocks, there are four main blocks; Network Management (NMT) manager, Service Data Object (SDO), Process Data Object (PDO) and Embedded MATLAB function. This paper describes the general concept Simulink setup and the explanation of applying CANopen protocol is given in the Embedded MATLAB Function codes.

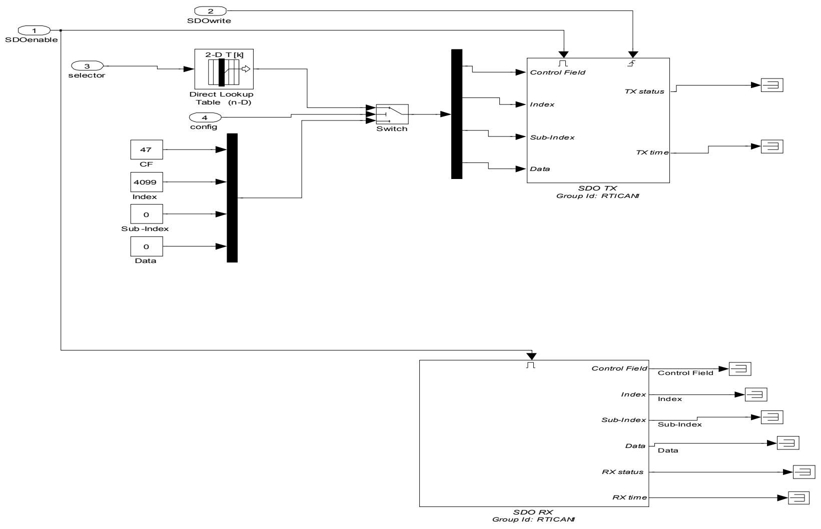

Fig 3. SDO manager

Table 1. Lookup Table

Message ID	Length	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	8	2F	00	18	02	01	00	00	00
601	8	2F	00	1A	00	00	00	00	00
601	8	23	00	1A	01	20	00	6C	60
601	8	23	00	1A	00	20	00	6B	60
601	8	2F	00	1A	02	02	00	00	00
601	8	2F	00	16	00	20	00	00	00
601	8	23	00	16	00	01	00	6B	20
601	8	2F	60	60	00	FE	00	00	00
601	8	2F	40	60	00	06	00	00	00
601	8	2F	40	60	00	0F	00	00	00

Basically, the SDO manager (see Figure 3) has 4 control signals; 'SDOwrite', 'SDOenable', 'selector' and 'config'. In general, the SDO manager is used to access stored data and to write the configuration data to a node's object dictionary which involved setting the device parameters, assign application data to PDOs and assign message identifiers. The data involved in this program are the configuration to run the Maxon motor according to the CANopen protocol. These data are stored into a look up table under the "SDO Manager" subsystem. In this pa particular test bench, the data have to be sent out 'automatically' using Embedded Matlab Function block.

SDO: used to initialise CANopen units, also defining Process Data Object.
PDO: are messaging used to send demands (e.g velocity) to each CANopen unit in a compact form. In particular, several data can be requested from each CANopen unit using a single message.
SDOwrite: To enable the SDO to write the data from the lookup-table.
SDOenable: To enable SDO operation.
Selector: The selector signal is an external counter to transfer the data lookup table in Table 1.

A.. NMT

In the CANopen NMT, the device implements a state machine which brings an internal initialisation every device in Pre-operational state. At this state, the node may be configured and parameterized via SDO. A NMT Slave is uniquely identified by its module ID. The NMT Manager device may switch all nodes to Operational state after writing data from the lookup table. At the NMT manager, there are two states of operation which are Pre-operational and Operational. Based on the CANopen protocol, the system begins with pre-operational mode at the beginning and the system will switch to the 'Operational' mode after completing writing the data at SDO manager through the lookup-table which leads to PDO enable.

B. Embedded MATLAB Function

Embedded MATLAB function block uses the textual language of the MATLAB software rather than Simulink block graphic language. In this embedded MATLAB function, it has all the states which involve with PDO, SDO and NMT manager.

The embedded Matlab Function activates the specific tasks according to the CANopen protocol. Initially, initialisation of variables involved in the code is carried out before entering a program statement. Then, the trigger signal acts as indicator to start the program code. According to the CANopen protocol, this program activates the `pre-operational' mode and continues to read each state according to the stored data (refer Table 1) in the lookup table. These states permit the initialisation of the CANopen slave node. After completing all states, the 'operational' mode is activated to run the motor according to the demand velocity.

4 Performance

A. Experimental Set Up

In order to realize the use of CANopen communication control, the following experimental set up has been used as seen in Figure 4. The underactuated finger is integrated with CANopen communication control in Matlab and Dspace.

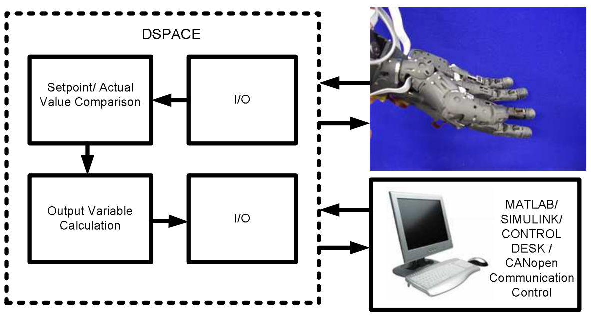
DSPACE
Setpoint/ Actual Value Comparison
I/O
Output Variable Calculation
I/O
MATLAB/ SIMULINK/ CONTROL DESK / CANopen Communication Control

Fig 4. Experimental Set Up

B. PD Controller and Result

We are testing PD (Proportional Derivative) controller in order to validate the use of CANopen communication control for underactuated system. The result shows that the PD controller can be successfully implemented by integrating the proposed CANopen communication control. This can be seen in Figure 5 where actual signal follows the demand signal satisfactorily.

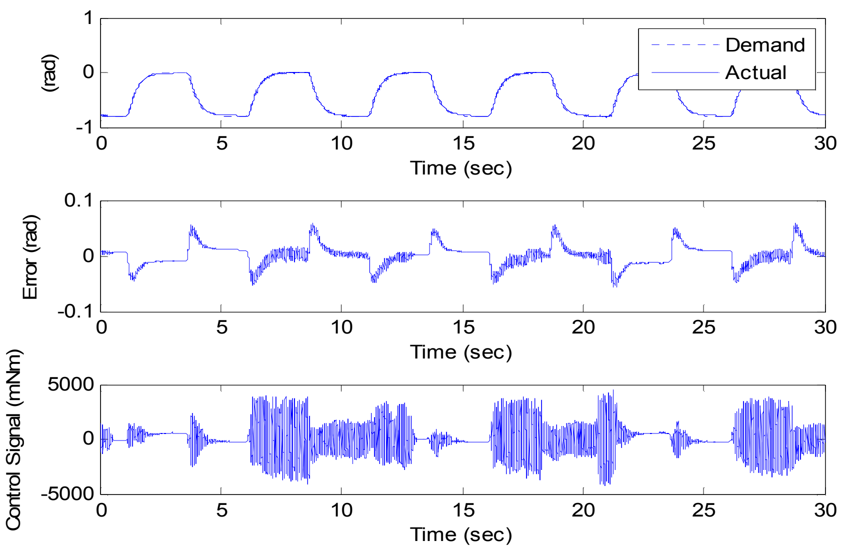

Fig 5. PD tracking performance

5 Conclusion

The implementation of a real-time embedded platform using CANopen protocol has been presented which incorporates a Matlab/Simulink environment and dSPACE DS1006. The values of the parameters and reference input signals of the CANopen protocol were stored in the lookup-table Simulink model. This platform has been validated on ControlDesk through dSPACE environment and tested on underactuated system. The experimental result shows that the actual signal follows the demand signal satisfactorily. Hence, the CANopen network is reliable and practical to be used in BERUL fingers.

References

1. J.-H. Oh, D. Hanson, W.-S. Kim, Y. Han, J.-Y. Kim, and I.-W. Park, “Design of android type humanoid robot albert hubo,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 1428–1433, Oct. 2006. April 1955.

2. T. Buschmann, S. Lohmeier, and H. Ulbrich, “Humanoid robot lola: Design and walking control,” *Journal of Physiology-Paris*, vol. 103, no. 3-5, pp. 141 – 148, 2009.I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in *Magnetism*, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
3. “Road vehicles – interchange of digital information controller area network (can) for high-speed communication,” in *Vehicle Power and* Propulsion Conference, 2008. VPPC’08. IEEE, vol. ISO 11898:1993, 1993.
4. F. Hartwich, B. Muller, T. Fuhrer, and R. Hugel, “Time triggered communication on can,” in *Proceedings 7th International CAN Conference*, 2000.
5. F. Hartwich, B. Muller, T. Fuhrer, and R. Hugel, “Timing in the ttcan network,” in *Proceeding 8th International CAN Conference*, Las Vegas 2002.
6. F. Hartwich, B. Muller, T. Fuhrer, and R. Hugel, “Fault tolerant ttcan networks,” in *Proceedings 8th International CAN Conference*, p. Las Vegas, 2002.
7. A. Pfeiffer, O. Ayre, “Embedded networking with can and canopen,” in *RTC Books*, vol. ISBN 0-929392-78-7, 2003.
8. F. Luo, Z. Chen, J. Chen, and Z. Sun, “Research on flexray communication system,” in *Vehicle Power and Propulsion Conference, 2008. VPPC’08. IEEE*, pp. 1–5, Sept. 2008.
9. G. Kopetz, H. Griinsteidl, “Ttp-a protocol for fault-tolerant real-time systems,” in *IEEE Computer*, pp. 14–23, January 1994.
10. “Canopen application layer and communication profile,” in *CAN in Automation CiA Draft Standard 301*, vol. 4.02, 2002.