Conveyor Belt Speed Control Through CAN BUS in CoppeliaSim using Arduino Mega2560

Sudip Chakraborty [1] & P. S. Aithal [2]
[1] D.Sc. Research Scholar, Institute of Computer Science and Information sciences, Srinivas University, Mangalore-575 001, India
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in
[2] ViceChancellor, Srinivas University, Mangalore, India
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Area of the Paper: Computer Science.
Type of the Paper: Experimental Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: https://doi.org/10.5281/zenodo.6415302
Google Scholar Citation: IJCSBE

How to Cite this Paper:
Chakraborty, Sudip, & Aithal, P. S., (2022). Conveyor Belt Speed Control Through CAN BUS in CoppeliaSim using Arduino Mega2560. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *6*(1), 194-201. DOI: https://doi.org/10.5281/zenodo.6415302.

International Journal of Case Studies in Business, IT and Education (IJCSBE)
A Refereed International Journal of Srinivas University, India.

Crossref DOI : https://doi.org/10.47992/IJCSBE.2581.6942.0159

Paper Submission: 08/03/2022
Paper Publication: 06/04/2022

Conveyor Belt Speed Control Through CAN BUS in CoppeliaSim using Arduino Mega2560

Sudip Chakraborty [1] & P. S. Aithal [2]
[1] D.Sc. Research Scholar, Institute of Computer Science and Information sciences, Srinivas University, Mangalore-575 001, India.
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in
[2] ViceChancellor, Srinivas University, Mangalore, India.
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *CAN Bus is a robust way to communicate inter-device Communication. Primarily it is used in the vehicle. But now, it is also used in various industrial automation fields. Among different branded robots, they use CAN-bus to control the robot movement. When the new researchers integrate the can bus into their project, they need to spend time running and handling the device through the CAN bus. This research work is basically for those trying to integrate the CAN bus into their project in a short period. Here, we briefly describe the CAN bus protocol first, then go through a practical experiment to better understand it. In our experiment, we feed the Analog value as a speed profile through the CAN bus to the CoppeliaSim virtual environment to see the speed variation of the simulated conveyor belt upon rotating the potentiometer. The complete project code and demo video are available on Github. The interested researcher can download and experiment on it.*

Design/Methodology/Approach: *This research work demonstrates how to exchange data through CAN bus between a virtual and a natural environment. We created a virtual environment using the CoppeliasSim robot simulator. Inside the simulator, we add a conveyor belt. Using c#, we created a bridge application that translates the data between the simulator and the natural environment outside it. We used two Arduino Mega 2560 boards and two MCP2515 based TJA1050 popular CAN driver modules. The CoppeliaSim requests data to the remote master through the bridge application and CAN bus. The remote master reads the analog value and sends it back to the simulator through the local master. CoppeliaSim changes its conveyor belt speed by parsing the data and converting it to the speed value.*

Findings/Result: *We can find some concepts and procedures to control the virtual elements from the external world through the CAN bus. We can convert any environmental analog parameter into digital form and display it locally or remotely inside the simulator. We can build the digital twins used to process monitoring in the industrial automation field following this experiment.*

Originality/Value: *Most CAN bus-related documents lack a practical approach or an accumulation of workable summaries. We demonstrate the CAN bus differently in this research work. This paper is an entirely practical-oriented demonstration. This is the unique approach to controlling the virtual element from the real world through the CAN bus. The new researcher can get helpful references to integrate the CAN bus into their project.*

Paper Type: *Experimental-based Research.*

Keywords: CAN bus, Control Simulator from the real world, Control over CAN bus, CoppeliaSim Conveyor belt demonstration

1. INTRODUCTION :

The CAN bus started its journey in 1983 at Robert Bosch GmbH. It was then finally released by the Society of Automotive Engineers in 1986. All controller devices can send the data over the bus in the CAN bus. Communicate over are required on the CAN network communication. A node may interface devices from simple digital logic to an extensive software embedded computer. Its main application

domain is automotive electronics. Now it is accepted by various domains due to its simplicity and robustness. It started using elevators, building automation, medical instrument metro railway, 3D printer, etc. Day by day, it continues to grab more and more fields. It makes wiring pretty simple. It is too effective where a huge sensor needs to connect with the controller. To join, only two wires are required, CAN-H and CAN-L. The wires are a twisted pair with a 120 Ω characteristic impedance.
When we were going to integrate the CAN bus into our project, lots of fragmented documents were found. Not only that, the practical approach is less prioritized in those documents. But we needed simple and tested practical examples to integrate inside our system. We researched it. And finally, we made the CAN bus functional in our project. Then we decided, whatever we worked on our project, we should document in a paper so that the new researcher could find some helpful guidance. Like our other projects [1-5], we used CoppeliasSim robot simulator to create virtual environment.

2. RELATED WORKS :

Renjun Li et. designed a test automotive CAN-controlled device. This system can display CAN frames (CAN 2.0A/B) received from or sent onto the CAN bus and record data on log files for offline evaluation. Users can configure several monitoring modes and CAN channel features with a PC application [6]. Ping Ran et al. developed a system using CAN bus controller SJA1000 and single-chip AT89C51. It has the advantages of long direct communication distances, high communication velocity, simple configuration, and low cost [7]. Li Ran et al., in their paper, works out an SAE J1939 application layer protocol to meet the system functional requirements and designs the software and hardware for the system. The method of software communication module includes CAN initialization unit, message sending unit, message receiving corps, and the interrupt service unit [8]. Taylor, A. et al. presents an algorithm that measures inter-packet timing over a sliding window. The average times are compared to historical averages to yield an anomaly signal. They evaluate this approach over a range of insertion frequencies and demonstrate the limits of its effectiveness [9]. Marchetti, M. et al., in their paper, proposes a novel intrusion detection algorithm. Its detection performance is demonstrated through experiments on real CAN traffic gathered from an unmodified licensed vehicle [10]. Q. Wang et. Al. Their work proposes a good security framework for a car over a CAN bus [11]. In their paper, M. Di Natale et al. describes an effective deadline encoding method and discusses its implementation and effects on the guaranteed analysis. Despite a limited processor overhead (less than 5% of CPU time), the proposed EDF implementation allows an increase (up to 20%) in the feasible network workload. This tradeoff will be more convenient as controller technology evolves [12]. Gaiden, M. et al. developed a security mechanism to protect the CAN. Their work does not require any modification in the standard procedure of the CAN bus nor to be implemented in each calculator of the network [13]. Bozdal, M. et al., in their paper, presents the CAN protocol and analyzes its security vulnerabilities. They survey the implemented attacks and proposed solutions in the literature [14]. Fugiglando *et al*. propose an unsupervised learning technique that clusters drivers in different groups and offers a validation method to test the robustness of clustering in a wide range of experimental settings [15].

3. OBJECTIVES :

This research provides some helpful information on who is struggling to integrate the CAN bus into their project. To work on the CAN bus, we need to know the basics of the CAN bus. For this, we introduce a minimal summary of it. For depth study lot of documents are available around the web. Some helpful link is provided in the recommended section. We use the popular Arduino board Mega2560 and MCP2515 based CAN bus driver modules TJA1050 for ease to arrange and experiment. We kept the experiment procedure simple for understanding better.

4. APPROACH AND METHODOLOGY :

At a glance, what about the CAN bus :

- ❖ The complete form of the CAN bus is Controller Area Network
- ❖ It is a two-wire robust vehicle bus standard designed for Communication among devices.
- ❖ Now it is used in various domains due to its simplicity and robustness
- ❖ Two types of CAN devices are present, CAN 2.0A uses an 11-bit identifier, and CAN 2,0B uses a 29-bit identifier.

- CAN is a multi-master serial bus.
- All nodes are connected through a two-wire bus, CANH and CANL. All nodes CANH joined. And same for CANL. The wires are a generally used twisted pair.
- Two states are present, dominant and recessive.
- Transmission speed depends on cable length. Below 40 meters, bit rates up to 1 Mbits/second, and 125 bits when the cable length is 500 meters.
- Each device consists of a unique device id.
- Figure 1 depicts the CAN bus base frame format

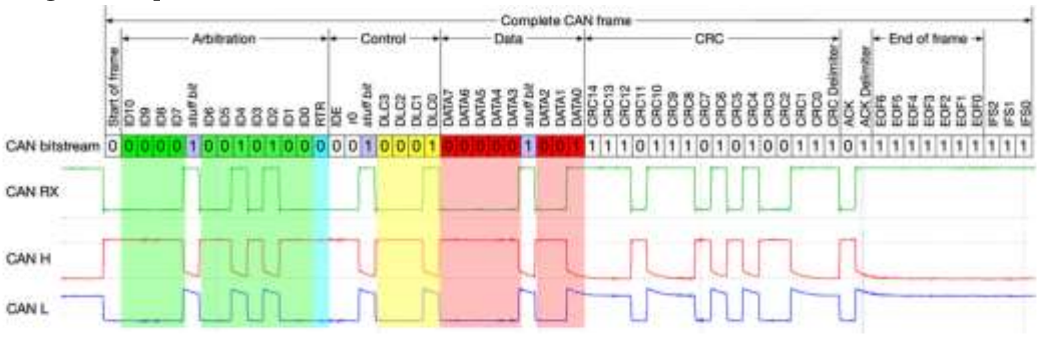


Fig. 1: Base frame format

- Figure 2 depicts the physical port of the CAN bus. The pin connection is below.
  Pin 2: CAN-Low(CAN-)
  Pin 3: GND (Ground)
  Pin 7: CAN-High(CAN+)
  Pin 9: CAN V+ (power)

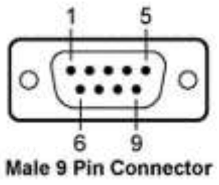


Fig. 2: CAN bus connector

- CAN has four frame types. The Data frame contains node data for transmission. The Remote frame includes information on a specific identifier. The Error frame includes information on an error. The Overload frame has a structure to inject a delay between data or remote frames.
- Figure 3 depicts the state machine of CAN driver 2515.

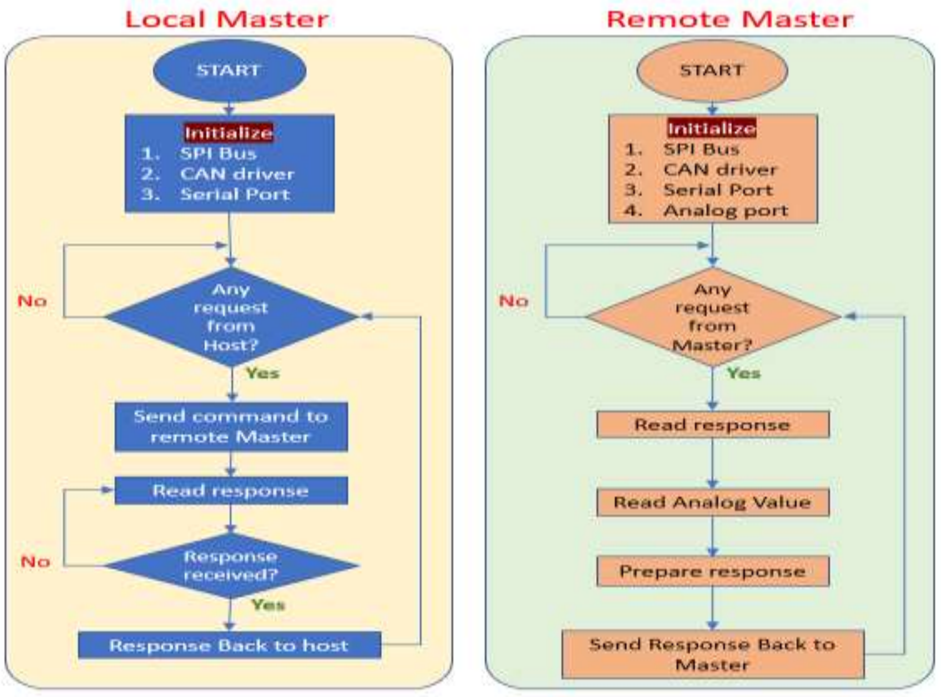


Fig. 3: Remote and local master state machine

Figure 3 depicts the machine's state of the local and remote nodes. After powering up, the local node initializes the SPI bus. If the SPI bus does not initialize first, the controller will not be able to communicate with the CAN driver. After properly initializing the SPI bus, the controller initializes the CAN driver. Then serial port needs to initialize. Otherwise, it cannot be able to listen to the command from the host (computer). The identical sequences must be followed to the remote node except the serial port. For remote node does not require a host interface. After initialization, both are ready to receive the command. When CoppeliaSim sends a conveyor belt speed data request, the host (computer) sends it to the local controller through the serial port. Receiving the command, it process and forward the request to the remote node through the CAN bus. The remote node instant purses the command and takes action. At first, it read the analog value through the analog port. Reading the data, create a response packet, and reply to the local controller over the CAN bus. The local controller reads the content and sends it back to the host application. The host application sends to CoppeliaSim over TCP/IP socket. One cycle is complete. It is repeated until the CoppeliaSim is running.

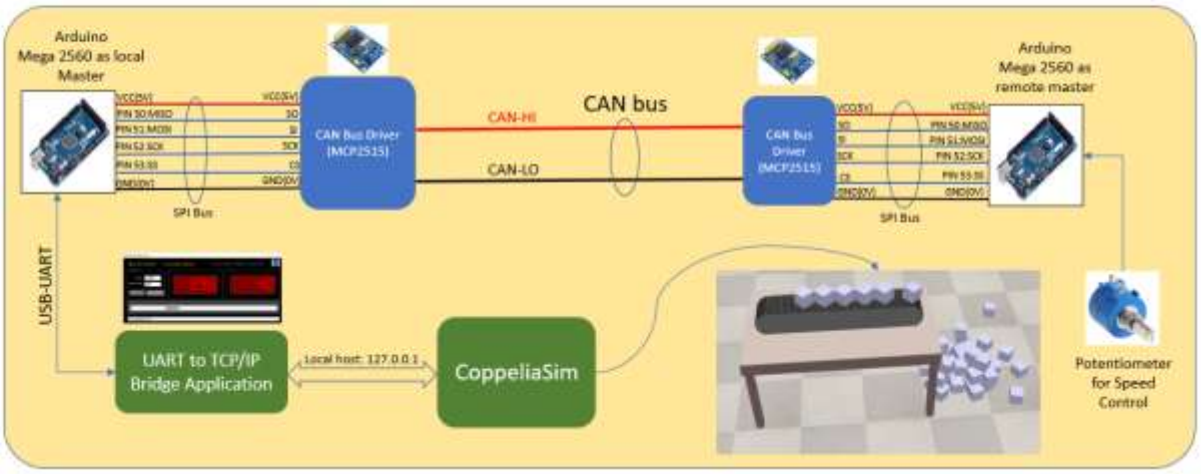


Fig. 4: The block diagram of our experiment setup

Figure 4 depicts the architecture of our experiment setup. We have taken two Arduino mega2560 and two MCP2515 CAN bus modules. In the figure, the right side Arduino acts as a remote node. One variable register of 1k/5k is connected with Analog pin A0. The board's SPI is associated with the CAN bus driver. On the left side, another CAN module is present. Each CAN-HI should be combined with the CAN-HI pin of others. The Same applies to CAN-LO. Here we can use any variant of Arduino board like Arduino NANO, UNO, DUE, etc., where at least one hardware SPI module is present.

5. EXPERIMENT :

Now we can do some experiments to understand CAN bus better. Below is the circuit diagram needed to follow to run the experiment (Figure 5).

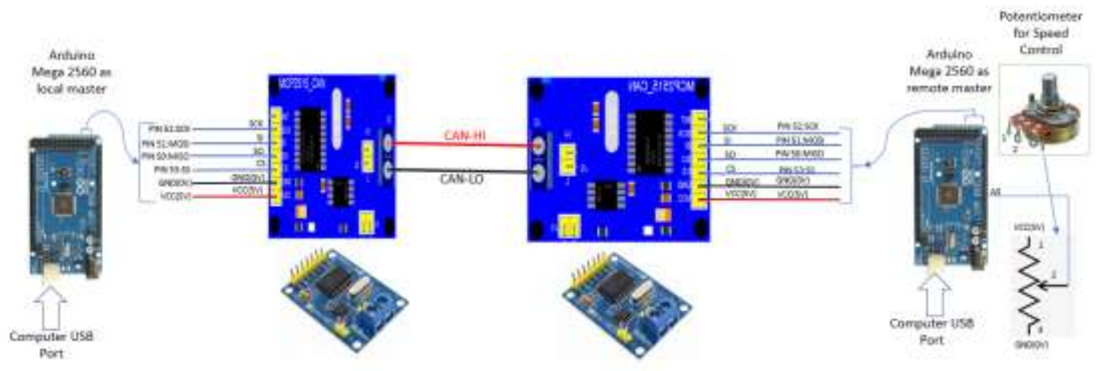

Arduino Mega 2560 as local master
MCP2515_CAN
CAN-HI
CAN-LO
Potentiometer for Speed Control
Arduino Mega 2560 as remote master
A0
VCC(5V)
GND(0V)
Computer USB Port

Fig. 5: The circuit diagram of the experiment

To run the experiment, we need to arrange the below materials. We can get it from online stores like Amazon, etc.

- Arduino Mega2560 with compatible USB Cables– 2 nos.
- MCP2515 Based module TJA1050- 2 nos.
- One 1k/5k potentiometer/Variable resistor – 1no.
- Few Male-Female breadboard hookup wires (around 12 nos.)
- Some wires for connection between two modules.

After collecting the materials, we need to connect like in figure 5. then we need to follow as below steps:

1) Download the project from Github.
2) Install Arduino IDE.
3) Take one Arduino 2560 and connect with Laptop/desktop Computer USB port.
4) Open Arduino IDE. From the downloaded project, Open the file under ARDUINO\CAN_MASTER_REMOTE\ ”CAN_Client.ino”. Upload the sketch (The upload procedure is available on the web).
5) Take another Arduino and upload ARDUINO\CAN_MASTER_LOCAL\” CAN_Master.ino”.
6) Now both power on.
7) Run PC Application\cBelt_Ctrl\bin\Debug\ cBelt_Ctrl.exe (might be installed some dot net component from the web to run the application).
8) Run CoppeliaSim simulator
9) Rotate the potentiometer/variable resistor.
10) Observe the Conveyor belt speed is changing on the rotation. Figure 6 shows our experiment setup. Figure 7 shows the bridge application, and figure 8 shows the conveyor belt running inside the CoppeliaSim simulator.

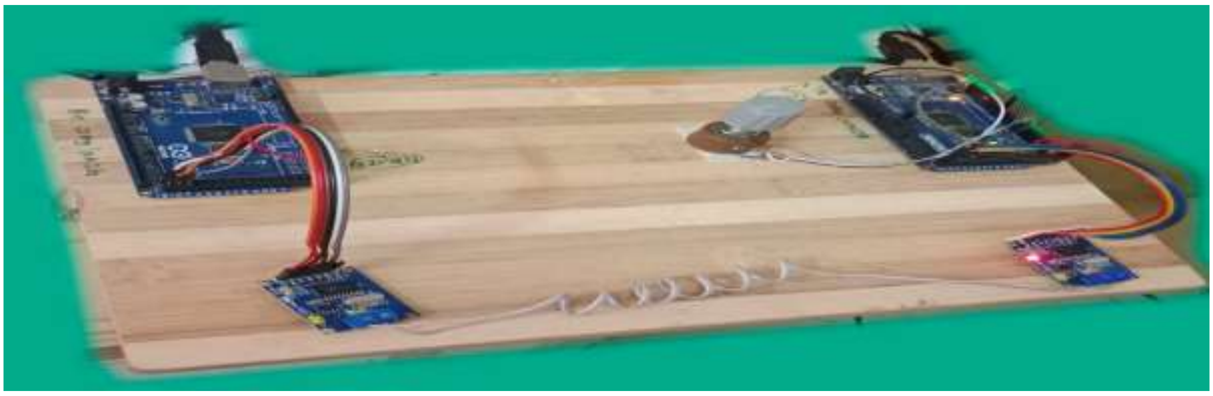
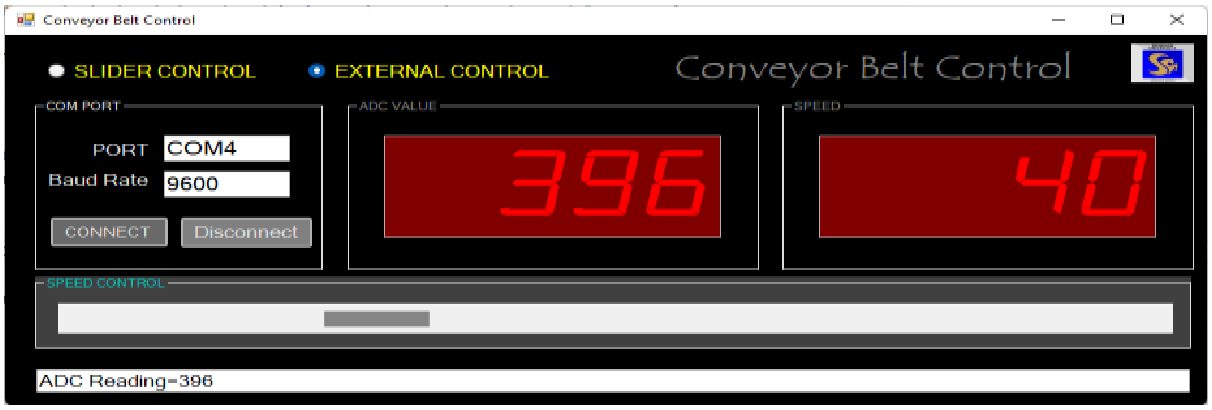


Fig. 6: Our experiment setup



Conveyor Belt Control

○ SLIDER CONTROL ● EXTERNAL CONTROL

Conveyor Belt Control

COM PORT
PORT COM4
Baud Rate 9600
CONNECT Disconnect

ADC VALUE
396

SPEED
40

SPEED CONTROL

ADC Reading=396

Fig. 7: Bridge application between the virtual and real world

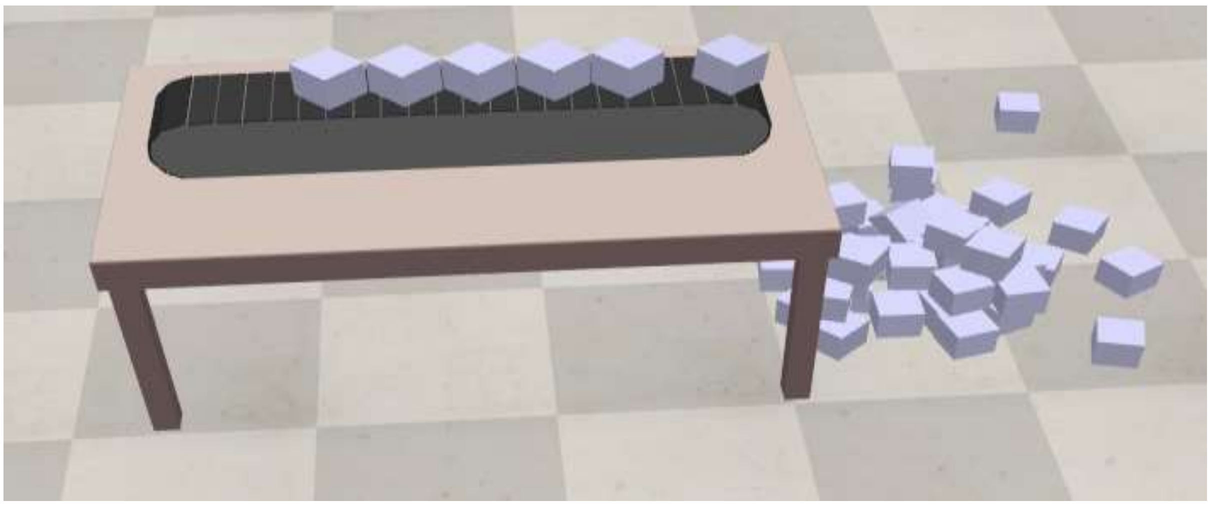


Fig. 8: The conveyor belt is rotating inside the CoppeliaSim

6. RECOMMENDATIONS :

- The project code is available from: https://github.com/sudipchakraborty/Conveyor-Belt-Speed-Control-Through-CAN-BUS-in-CoppeliaSim-using-Arduino-Mega2560
- CoppeliaSim can be downloaded from: https://coppeliarobotics.com/downloadsv
- Visual Studio 2022 community edition can be downloaded from: https://visualstudio.microsoft.com/downloads/
- CAN driver chip datasheet link:- https://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf
- Latest Arduino IDE download link: https://www.arduino.cc/en/software
- For details description of the CAN bus can be found at https://en.wikipedia.org/wiki/CAN_bus
- Arduino code adapted from: https://forum.arduino.cc/t/read-line-from-serial/98251/4
- A good tutorial on CAN bus: https://www.kvaser.com/can-protocol-tutorial/
- For better understanding, we capture a live demo and compress it using the online converter tool https://www.veed.io/video-compressor. The video is available in the Github link.
- All parameters are not calibrated. We need to calibrate for a more accurate result.
- This application is not entirely bug-free. We debugged what we got at experiment time. May persists several bugs, which can be debugged under more tests.
- We developed core functions only. The researcher can add additional functionality for better usability.

7. CONCLUSION :

In this research work, we experimented with CAN bus, which is now defacto standard for multi-master, robust Communication between devices. We introduced CAN bus basics and observed the practical implementation procedure. We fetched the data from the remote node through the CAN bus and processed it by bridge software written in C#. The processed data is received by CoppeliaSim virtual robot simulator. As a result, the Conveyor belt speed changes when the potentiometer changes. This work may be a reference document for the researcher if anyone is stuck on implementing the CAN bus project.

REFERENCES :

[1] Chakraborty, S., & Aithal, P. S. (2021). Forward Kinematics Demonstration of 6DF Robot using CoppeliaSim and C. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 29-37. Google Scholar↗

[2] Chakraborty, S., & Aithal, P. S. (2021). A Custom Robotic ARM in CoppeliaSim. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 38-50. Google Scholar

[3] Chakraborty, S., & Aithal, P. S. (2021). An Inverse Kinematics Demonstration of a Custom Robot using C# and CoppeliaSim. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *5*(1), 78-87. Google Scholar

[4] Chakraborty, S., & Aithal, P. S. (2022). A Simulated 3D Printer in CoppeliaSim. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *6*(1), 22-32. Google Scholar

[5] Chakraborty, S., & Aithal, P. S. (2021). Forward and Inverse Kinematics Demonstration using RoboDK and C. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 97-105. Google Scholar

[6] Renjun Li, Chu Liu, and Feng Luo, (2008). A design for automotive CAN bus monitoring system. *IEEE Vehicle Power and Propulsion Conference*, *2008*(1). 1-5. DOI: 10.1109/VPPC.2008.4677544. Google Scholar

[7] Ping Ran, Baoqiang Wang, and Wei Wang, (2008). The design of communication converter based on CAN bus. *IEEE International Conference on Industrial Technology*, *2008*(1), 1-5. DOI: 10.1109/ICIT.2008.4608607. Google Scholar

[8] Li Ran, Wu Junfeng, Wang Haiying, and Li Chechen. (2010). Design method of CAN-BUS network communication structure for an electric vehicle. *International Forum on Strategic Technology 2010*(1), 326-329. DOI: 10.1109/IFOST.2010.5668017. Google Scholar

[9] Taylor, A., Japkowicz, N. and Leblanc, S. (2015). Frequency-based anomaly detection for the automotive CAN bus. *World Congress on Industrial Control Systems Security, 2015*(1), 45-49. DOI: 10.1109/WCICSS.2015.7420322. Google Scholar

[10] Marchetti, M. and Stabili, D. (2017). Anomaly detection of CAN bus messages through analysis of ID sequences. *IEEE Intelligent Vehicles Symposium, 2017*(4), 1577-1583. DOI: 10.1109/IVS.2017.7995934. Google Scholar

[11] Wang, Q. and Sawhney, S. (2014). VeCure: A practical security framework to protect the CAN bus of vehicles. *International Conference on IoT*, 2014(1), 13-18. DOI: 10.1109/IOT.2014.7030108. Google Scholar

[12] Di Natale, M. (2000). Scheduling the CAN bus with earliest deadline techniques. *Proceedings 21st IEEE Real-Time Systems Symposium*, 2000(1), 259-268. DOI: 10.1109/REAL.2000.896015. Google Scholar

[13] Gaiden, M., Gaiden, M. H. and Trabelsi, H. (2016). An intrusion detection method for securing in-vehicle CAN bus. *17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), 2016*(1), 176-180. DOI: 10.1109/STA.2016.7952095. Google Scholar

[14] Bozdal, M., Samie, M., and Jennions, I. (2018). A Survey on CAN Bus Protocol: Attacks, Challenges, and Potential Solutions. *International Conference on Computing, Electronics & Communications Engineering (access), 2018*(1), 201-205. DOI: 10.1109/iCCECOME.2018.8658720. Google Scholar

[15] Fugiglando, U., et al., (2019). Driving Behavior Analysis through CAN Bus Data in an Uncontrolled Environment. *IEEE Transactions on Intelligent Transportation Systems*, *20*(2), 737-748. DOI: 10.1109/TITS.2018.2836308. Google Scholar

****