OULU UNIVERSITY OF
APPLIED SCIENCES

Tuukka Luoma

DEVELOPMENT OF INTELLIGENT WHEELCHAIR

DEVELOPMENT OF INTELLIGENT WHEELCHAIR

Tuukka Luoma
Bachelor’s Thesis
Spring 2013
Degree Programme in Automation Engineering
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Automation Engineering

Author(s): Tuukka Luoma
Title of thesis: Development of Intelligent Wheelchair
Supervisor(s): Tero Hietanen
Term and year when the thesis was submitted: Spring 2013
Number of pages: 58 + 14 appendices

The purpose of this thesis was to develop an intelligent wheelchair using a 9-axis motion sensor. The goal was to create a control system that allows a user of an electric-powered wheelchair control it hands-free.

Used devices were installed and configuration was done. A control system was built using MATLAB and its Simulink environment. The control system used fuzzy logic controllers to imitate the behaviour of a joystick controlled electric-powered wheelchair. The fuzzy control system bypassed the joystick control and it controlled the motors of the wheelchair directly.

The result of this project was a control system that is capable of controlling a wheelchair using body movements of its user. It provides a platform for further development and tuning of the developed control system.

Keywords:
Wheelchair, motion control, motion sensor, fuzzy control, MATLAB, simulink

ACKNOWLEDGEMENTS

This Bachelor’s Thesis was commissioned by Kumamoto National College of Technology (KNCT). The supervisor from KNCT was Professor Hirofumi Ohtsuka from Department of Control and Information Systems Engineering. The supervisor from Department of Automation Engineering of Oulu University of Applied Sciences was Senior Lecturer Tero Hietanen.

I would like to thank Professors Hirofumi Ohtsuka and Kousei Nojiri for their invaluable help throughout this project. A thank you to Senior Lecturer Tero Hietanen for helping bringing the documentation of the project into a closure. I would also like to thank English teacher Marjo Heikkinen who helped with the layout and grammar of this thesis.

I would like to say thanks also to my wife Yuka who supported me throughout the difficulties I faced while writing this documentation.

May 11th, 2013

Tuukka Luoma

CONTENTS

TERMS AND ABBREVIATIONS

A	Ampere, the SI unit of electric current
Ah	Ampere-hour, a unit of electric charge
CAN	Controller Area Network
CANopen	A communication protocol used in automation
CiA	CAN in Automation
deg/s	Degree per second, an unit of rotational speed
EPW	Electric-Powered Wheelchair
G	Gauss, an unit of magnetic flux density in magnetic field B
g	An unit of g-force measured by accelerometer
g_n	Standard gravity, defined as 9.806 65 m/s^2
GUI	Graphical User Interface
IMU	Inertial Measurement Unit
IMU-Z	ZMP IMU-Z, a 9-axis motion sensor
I/O	Input/Output
KNCT	Kumamoto National College of Technology
Mbit/s	Megabit per second, a unit of data transfer rate
M-file	MATLAB source code file
NiMH	Nickel-metal hydride, a type of battery
PLC	Programmable logic controller
RS-232	A series of standards commonly used in computer serial ports
USB	Universal Serial Bus, a standard for connecting computer peripherals
V	Volt, the SI unit for electric potential difference
ZMP	ZMP INC., a Japanese robot company that developed the IMU-Z sensor

1 INTRODUCTION

A wheelchair is a chair designed for people who need an aid to replace walking. Reason for such need can be either illness or disability that prevents a person from walking normally. There are several different kinds of wheelchairs depending on their purpose and the way they are powered. In this thesis the main focus is on the electric-powered wheelchair (EPW).

The main goal for this thesis was to develop an intelligent hands-free wheelchair that is controlled by the movements of the wheelchair user. In this thesis the movements were tracked with a 9-axis motion sensor that was attached to the body of the wheelchair user. Kumamoto National College of Technology (KNCT) has developed different types of intelligent hands-free wheelchairs in the past such as voice and pressure mat operated systems.

This thesis presents one of the many possible ways to develop a control system for an electric-powered wheelchair. It is by no means the best solution but given the time frame and resources it gave acceptable results and a platform for further development of the system.

The project was executed with MATLAB and its Simulink environment. MATLAB is a widely used platform for building control systems and it gives easy and versatile possibilities to tune a system. Because of this reason MATLAB was chosen as the platform of the project to be built upon.

2 METHODS TO DEVELOP INTELLIGENT WHEELCHAIR

In this section the theory and techniques to build an intelligent wheelchair are taken a closer look at.

2.1 WHEELCHAIR

A wheelchair is a device designed to help people who need a replacement for walking. The need for this can be caused by a illness, injury or disability. Wheelchairs come in different kinds and they can be categorised for example by design, control method and use.

Most wheelchairs share the same basic design that contains a seat, foot rests, two small caster wheels in the front and two large wheels in the back. Many wheelchairs also incorporate handles in the back for a helper to push the wheelchair from the back.

The main difference between the early wheelchairs and the modern wheelchairs is the chassis. The first wheelchairs were rigid chairs as the mechanical know-how at the time was not up to present day standards yet. Modern wheelchairs come in two major designs: folding or rigid. Folding chairs are a good option if the chair needs to be put in a storage. A folding chassis makes it possible for the chair to fit in much smaller space that rigid chairs. On the other hand rigid chairs also come often with means to dismantle the chair in order to fit it in a car for transit. Also the materials are vastly different now. The early wheelchairs from pre-20th century were mainly built from materials from nature such as wood. The chassis of a modern wheelchair today is made of metal. Especially the rigid chairs incorporate lightweight materials like aluminium and titanium. These materials also make the wheelchairs very durable.

A more relevant way for this thesis to categorise wheelchairs is done based on how the chair is powered. Manually powered wheelchairs require a human to move the wheelchair. This can be done by the user or in some cases also a helper can move the wheelchair by pushing it from the back. The most common way to move a manually powered wheelchair is for the user to turn the wheels by pushing the wheels forward using handrims attached to outer rims of the back wheels. In some cases also a foot propulsion can be used. This happens when the user has limited hand movement. There are many variants of hand and foot propulsion that incorporate either propulsion with one or two limbs.

A modern electric-powered wheelchair can be seen in figure 1 below.

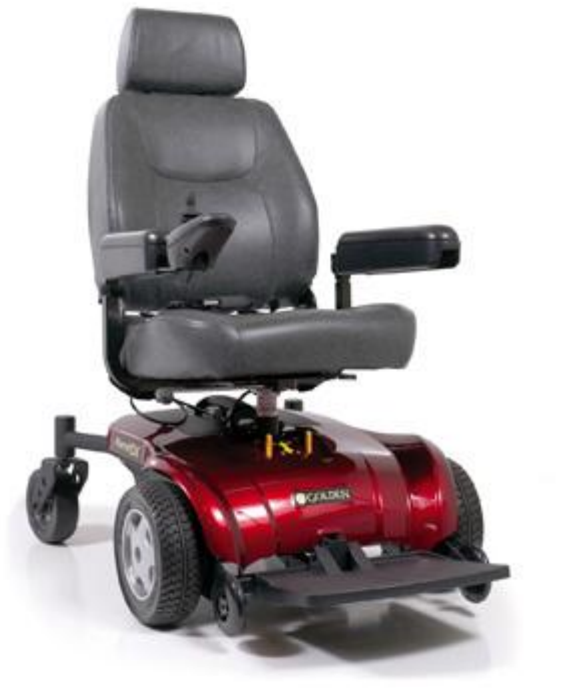


Fig. 1. A modern electric-powered wheelchair[1]

Electric-powered wheelchairs (EPW) are moved by electric motors and a navigational control system. Speed of an EPW is usually 6-10 km/h. The most common control system is a joystick attached to the armrest. If the user cannot use an armrest mounted joystick, there are other options such as chin-operated joysticks and sip-and-puff sensors. Also other special control methods can be used such as motion control. The electric motors are usually powered by 4 or 5 A rechargeable batteries.[2]

2.2 MOTION TRACKING

Motion tracking is a method used since the 1970s to track movement and make digital models based on the tracked data. The movements are recorded in high sample rate. Only the movements of an actor are recorded, not the visual appearance. The movement data is then processed into a model. This model performs the same movements as the human actor. It has many uses in military, entertainment and robotics. For example motion tracking is commonly used in movie industry these days to make animated characters using recorded movements of a human actor.[3]

There are several different ways to do motion tracking and each one has its own advantages and shortages. Unfortunately there is no perfect single method to do motion tracking. However, different methods can still give good results on specific tasks. The main advantages and shortages are listed in table 1.

Table 1. Advantages and shortages of different motion tracking methods.[4]

Sensing method	Advantages	Shortages
Acoustic	+Range compared to some other sensing methods	-Accuracy is affected by conditions (wind, temperature, humidity, etc...) -Low measure rate due to 5...100 ms delay caused by multi-path -Line-of-sight required
Inertial	+Self-contained +No line-of-sight required +Resistant to noise and electromagnetic field +Low latency (≤ 5 ms) +High measure rate +1000's of sample per second +Very low jitter	-Bias error of accelerometer causes drift (4.5 meters after 30 seconds)
Magnetic	+Small size +No line-of-sight required +Multiple sensors excited by a single source	-Limited range -Conductive material (changes the shape of the magnetic field)
Mechanical	+Precise	-Small range
Optical	+High spatial precision +High measure rate	-Line-of-sight required
Radio and microwave		-Equipment big and expensive -Cannot be used indoors due to multi-path problems

2.3 INERTIAL MEASUREMENT UNIT

An Inertial Measurement Unit, or IMU for short, is a device designed to measure velocity, orientation and gravitational forces of a craft of some kind. Common uses for IMU are in aircrafts and different kinds of spacecrafts. Sometimes IMUs are used in motion capture work but a more well-known example of an IMU in use is the self-balancing Segway Personal Transporter. The IMU works by detecting changes in acceleration and rotation using its accelerometers and gyroscopes.

An IMU is an electronic device that usually incorporates several accelerometers and gyroscopes. In some cases there can also be additional sensors. The sensors are placed in orthogonal pattern so that each accelerometer and gyroscope detects changes on a single axis in a three dimensional space. Figure shows the construction of the sensor placements in an IMU.[5]

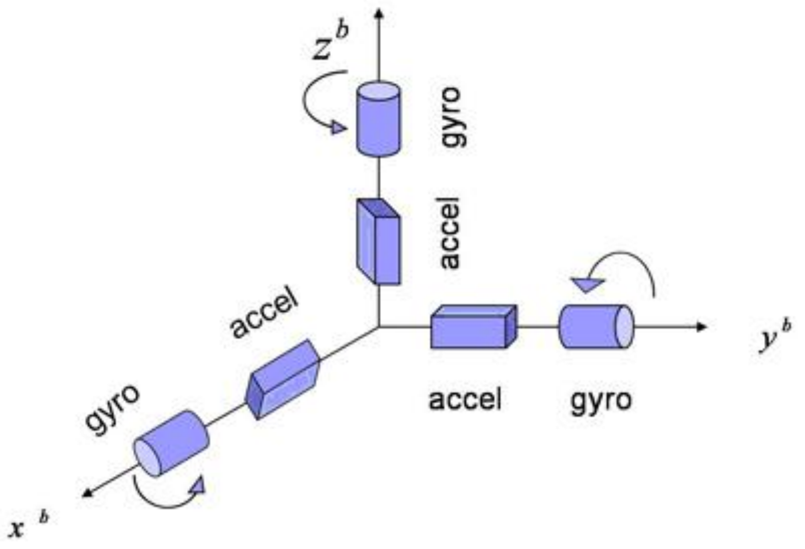


Fig. 2. Placements of accelerometers and gyroscopes in an IMU[6]

2.4 CONTROLLER AREA NETWORK

Controller Area Network (CAN) was originally designed for the needs of the car industry. Because of this starting point the interference resistance and the reliability of data transfer were the main objectives when CAN was developed. In each CAN-message there is a checksum that is used to make sure that the message is delivered in its original form. In addition to this CAN-bus is a serial bus which means increased resistance for interference. In the beginning of a CAN-message there is an ID-field that helps identifying commands and signals that are sent to different devices. All messages that are moving in CAN-bus can be read by all devices. In other words there is no need to send the same information more than once. Moreover this makes diagnostics easy because diagnostics device will not affect other devices.[7, p.2]

CAN-bus is a multi-master bus where each node can send a message through the bus on its own initiative. Each message can contain up to 8 bytes (64 bits) of information. The messages are broadcasted to all nodes but only the nodes that need the information are to receive the message. The sender is not sending the message to any node specifically but the sent message is available to all nodes. CAN does not use addresses for communication. This means that in principle there is no set maximum number of nodes on the bus. In practice transceiver chips always set a maximum number of nodes. Maximum transfer rate of CAN-bus is 1 Mbit/s when the bus is no longer than 40 meters. [8, p.4-5]

CAN-bus can be used in many kinds of environments. These include busses, elevators, robots, measuring equipment and PLC-units. CAN-bus can be fitted in almost any kind of device as long as the data transfer range is relatively short and the objective is a real-time communication between controllers. Because of its limitations in transfer rates CAN-bus is not the ideal method when big quantities of data has to be transferred, for instance video stream.[8, p.1]

One could describe CAN-bus as a network of processors as opposed to traditional sense of an industrial computer network. Regardless of this CAN-bus is often used like other industrial network systems.[8, p.2]

2.4.1 Physical Layer

The purpose of the physical layer in CAN-bus is the data transfer between the different devices on the hardware level. The layer can be divided in three parts as shown in table 2. A wiring structure of CAN-bus is presented in figure 3.

Table 2. Three parts of physical layer in CAN-bus[7, p.3]

Formation of the physical signal	Bit encoding and decoding Bit timing Synchronization Message filters by hardware
Connection of the physical transmission path	Properties of transmitter and receiver
Transmission path dependent connection	Cables Connectors

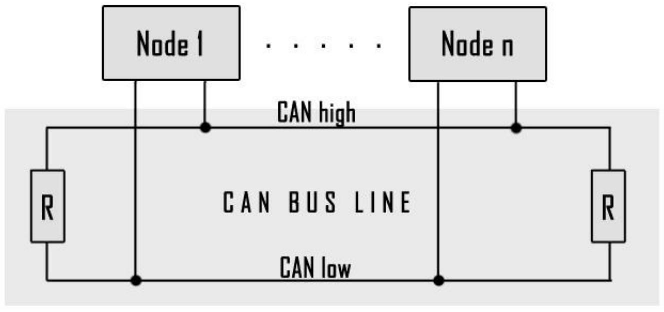


Fig. 3. Wiring of CAN-bus

In the figure 3 one can see how the devices (nodes) are connected in parallel to a two-wire cable. At the ends of the pair-wire cable there are termination resistors to prevent signals from refraction. The devices are observing the voltage in the bus. This voltage represents two signals: CAN low (bit value 0) and CAN high (bit value 1).

The physical maximum length of CAN-bus is limited by the delay that increases as the bus gets longer. The table 3 shows the maximum lengths of a bus for some transfer rates.

Table 3. Maximum lengths of CAN-bus for different data transfer rates[7, p.4]

Data transfer rate	Maximum length of bus
1 Mbit/s	30 m
800 kbit/s	50 m
500 kbit/s	100 m
250 kbit/s	250 m
125 kbit/s	500 m
62.5 kbit/s	1 km
20 kbit/s	2 km
10 kbit/s	5 km

When designing CAN-bus it is important to take notice of the number of nodes. As the number of nodes increases also both capacitance and resistance increases. This interferes the identification of signals. It is possible to make the CAN-bus longer by using repeaters that amplify the moving signals. On the other hand, repeaters can add some delay to the bus so in some cases it is necessary to decrease the data transfer rates.[7, p.4]

2.4.2 Frame Formats

CAN-devices support two different kinds of data frame formats; base frame and extended frame. The main difference between the two formats is the length of the ID-field. The standard frame has an ID-field of 11 bits and the extended frame has 29 bits in its ID-field. The ID-field contains information such as the type of message and which device the message is intended to. The extended frame offers more possibilities for message types and device addresses. Despite this the extended frame is not as widely used as the standard frame. For example the popular higher level protocol CANopen, that is based on CAN, uses the standard frame.

Typical data frame formats of the base frame and extended frame can be seen in appendix 1.

In addition to data frames there are also remote frame, error frame and overload frame. The purpose of the remote frame is to request a transmission of a specific ID. Error frame is transmitted by a device that detects an error. Overload frame causes delay between data and remote frames to prevent overloading of transmissions.[9]

2.5 FUZZY CONTROL SYSTEM

Fuzzy logic is a mathematical method that compares analog input signals to pre-determined logical variables, membership functions or “fuzzy sets”, that correspond to values between 0 and 1. The main difference between traditional digital logic is that in fuzzy logic values between 0 and 1 are allowed as in the purely digital logic only acceptable values are 0 and 1.

Fuzzy logic is used in control system to imitate a human-like decisions in control. While many different control methods can give a good performance in certain tasks, the main advantage of fuzzy logic is that the problem and solution are often easy to be described in a way that human operators can understand.

Fuzzy sets are the brain of a fuzzy control system. Fuzzy sets are in charge of converting analog input values into scale of 0 to 1. This new value of 0 to 1 describes how strong measured input corresponds to pre-determined membership functions. This whole process that fuzzy sets do is called “fuzzification”.[10]

Fuzzy logic could be used in an everyday situation that does not have an absolute truth but many truths. A common example is how to describe a temperature using fuzzy logic. Below in figure 4 is showed how a temperature could be expressed in fuzzy logic membership functions.

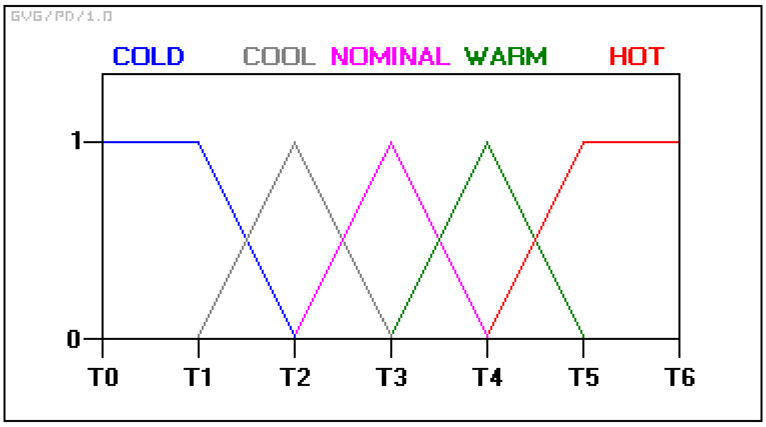


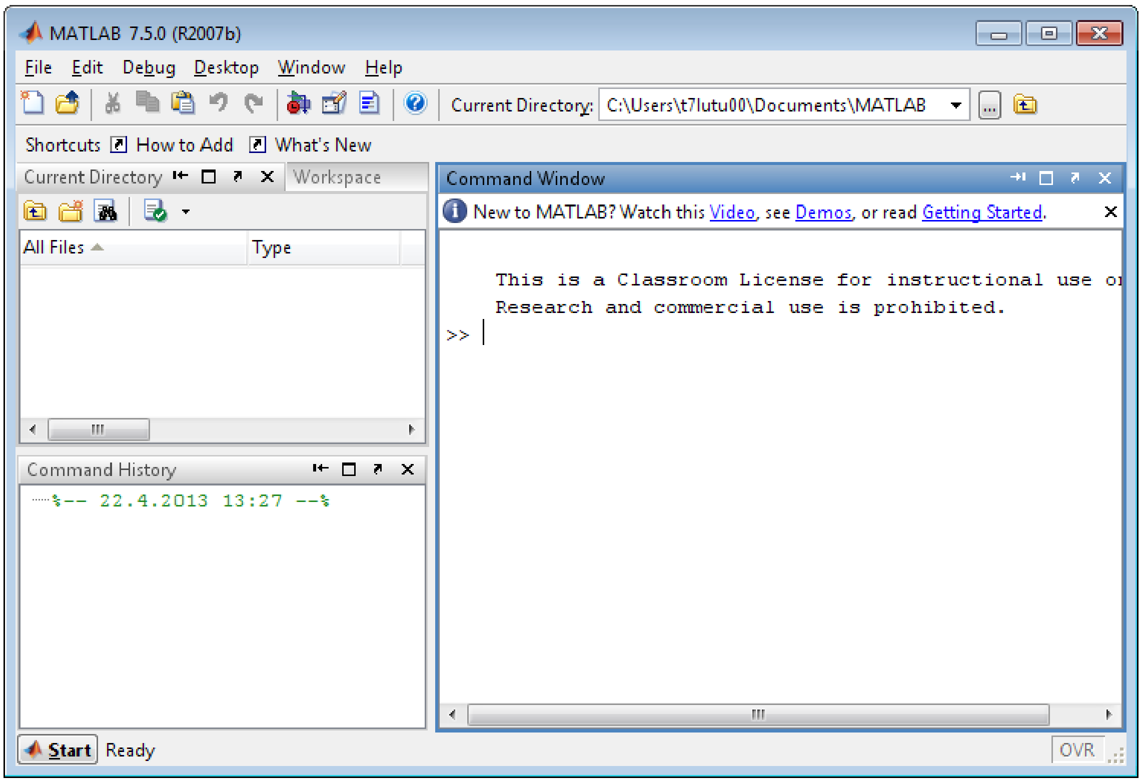
Fig. 4. Fuzzy logic membership functions to represent temperature[11]

The figure above can be translated into human language very easily. For example temperature T1 could be +10 °C and T2 could be +15 °C. Temperature of +12.5 °C has equal memberships of both “cold” and “cool” functions. Therefore it could be expressed as “pretty cold”. Likewise “pretty hot” would take place somewhere between temperatures T4 and T5.

2.6 MATLAB

MATLAB is a software environment for numerical programming developed by MathWorks. It is a versatile tool for matrix manipulations, plotting of functions and data, creation of user interfaces, simulation of control systems and many other applications used in engineering.

An overview of MATLAB can be seen in figure 5.


MATLAB 7.5.0 (R2007b)
File Edit Debug Desktop Window Help
Current Directory: C:\Users\t7lutu00\Documents\MATLAB
Shortcuts How to Add What's New
Current Directory Workspace
All Files Type
Command History
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Start Ready
OVR

```
  This is a Classroom License for instructional use o
  Research and commercial use is prohibited.
>> 
```

```
%-- 22.4.2013 13:27 --%
```

Fig. 5. An overview of MATLAB

2.6.1 Simulink

Simulink is a block diagram environment that is integrated to MATLAB. It is a graphical editor that allows building, controlling and simulating continuous-time and discrete-time systems. It comes with libraries of blocks that are used to simulate control systems in graphical editor.

Some of the block libraries in Simulink are shown in figure 6. Simulink comes with many blocks suitable for numerous different purposes in control system designing.

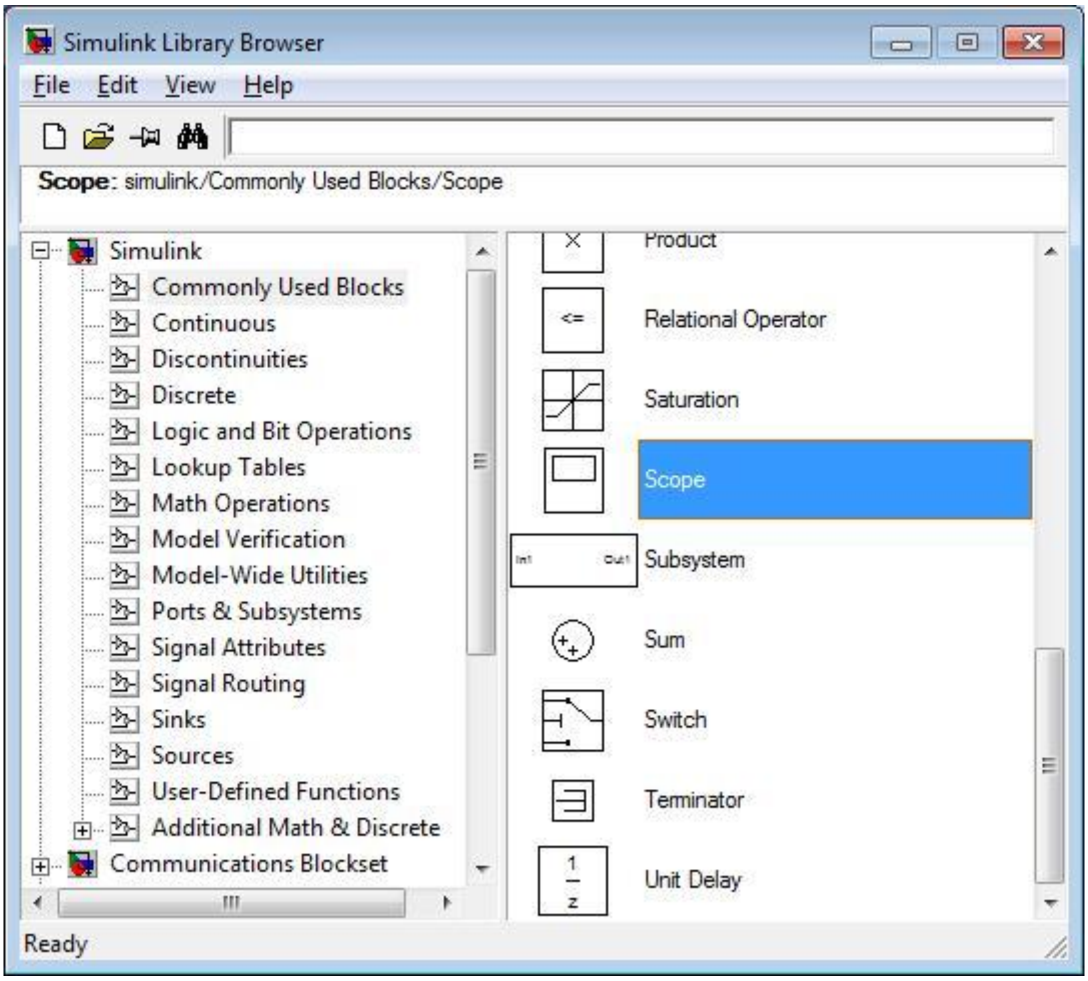


Fig. 6. Simulink Library Browser

Simulink is a graphical tool that requires little to none knowledge of programming. Only knowledge of how the simulated system works is needed. Below in figure 7 there is an example Simulink model of a PID control system created in Simulink.

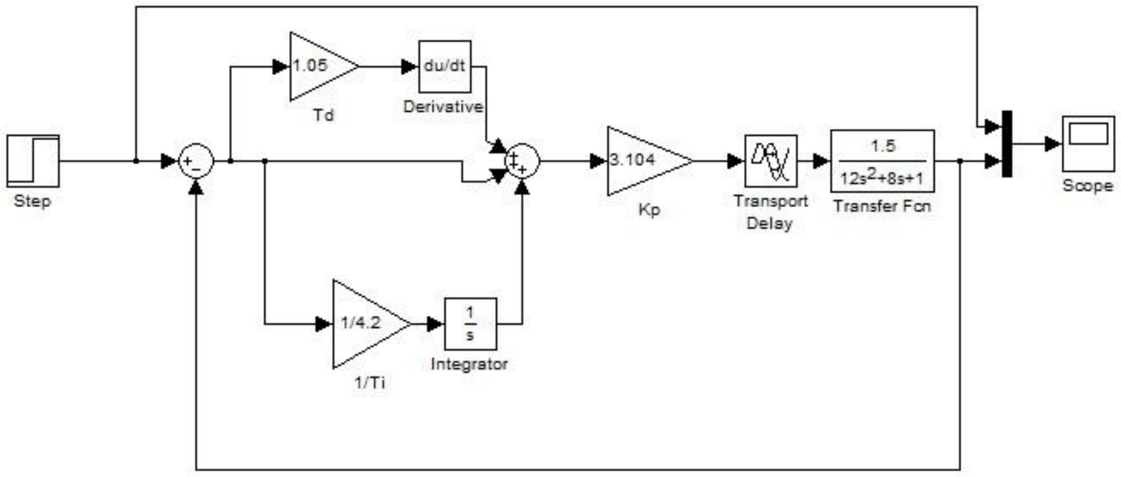
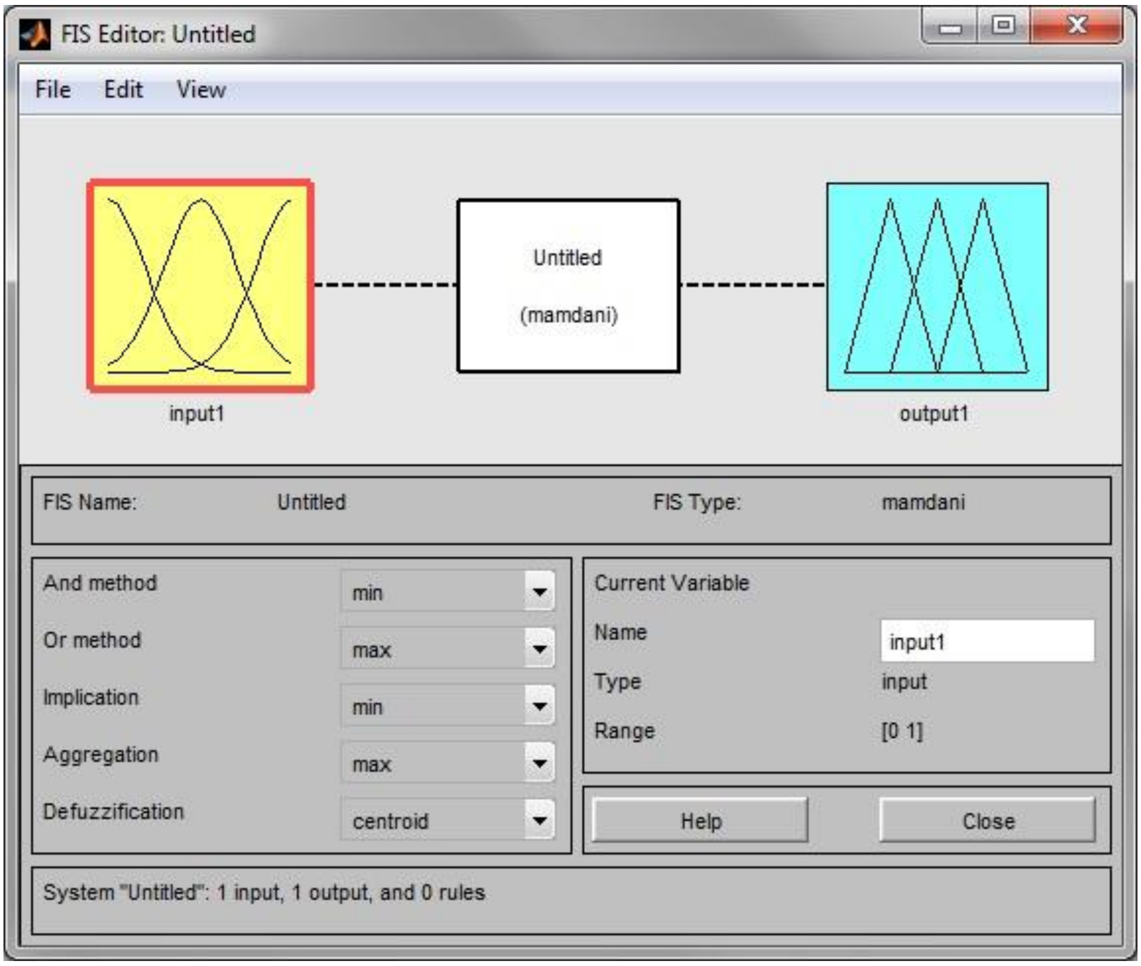


Fig. 7. A PID control system in Simulink environment

2.6.2 Fuzzy Logic Toolbox

Fuzzy Logic Toolbox is not required for designing fuzzy logic systems in MATLAB but it provides tools that make building fuzzy systems easy. The two main additions to MATLAB in this toolbox are FIS Editor and fuzzy logic Simulink blocks. The FIS Editor is shown in figure 8 below.



FIS Name: Untitled
FIS Type: mamdani

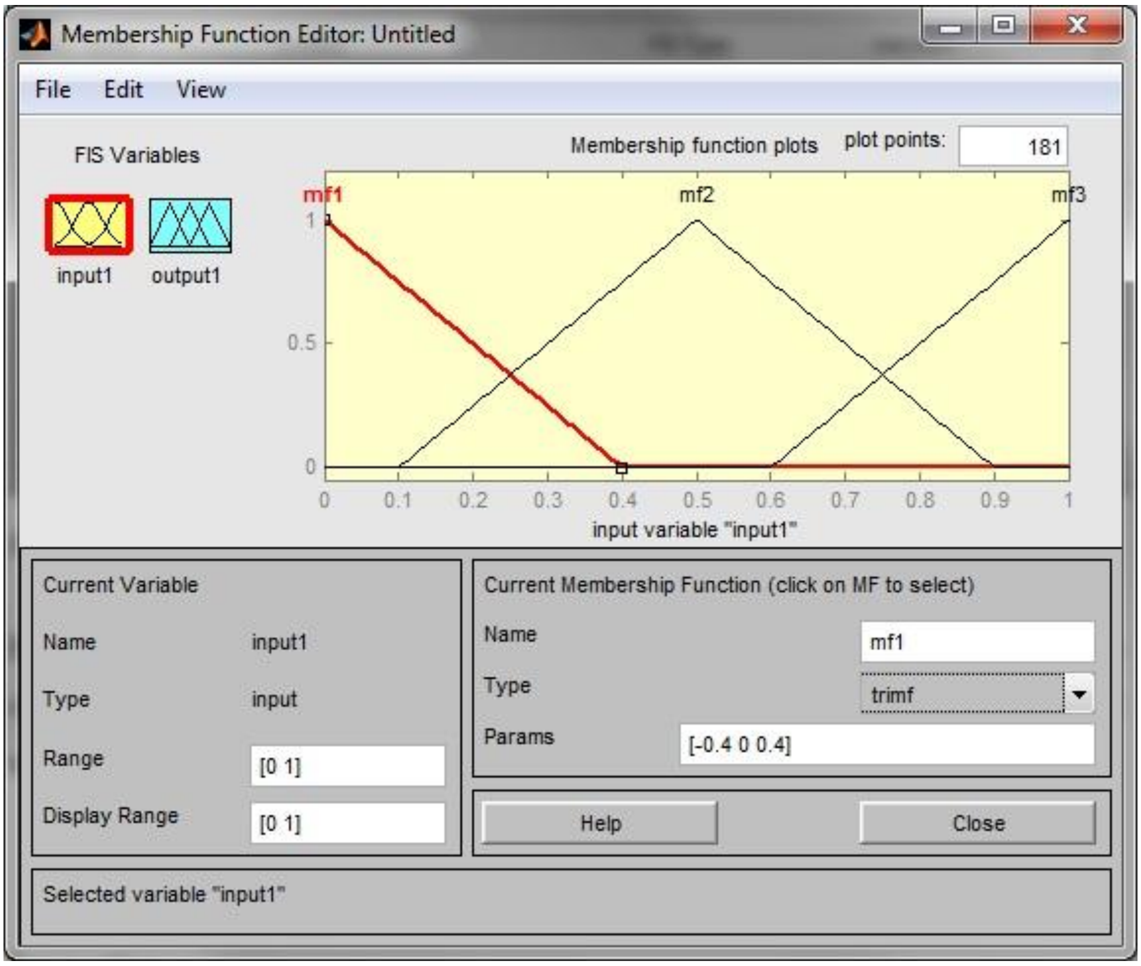
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: input1
Type: input
Range: [0 1]

System "Untitled": 1 input, 1 output, and 0 rules

Fig. 8. FIS Editor

FIS Editor is used for designing fuzzy logic controllers. Input and output signals are determined by choosing the desired shape and number of membership functions in Membership Function Editor (seen in figure 9). Also the range of signals can be changed in Membership Function Editor.



Current Variable

Name input1

Type input

Range [0 1]

Display Range [0 1]

Current Membership Function (click on MF to select)

Name mf1

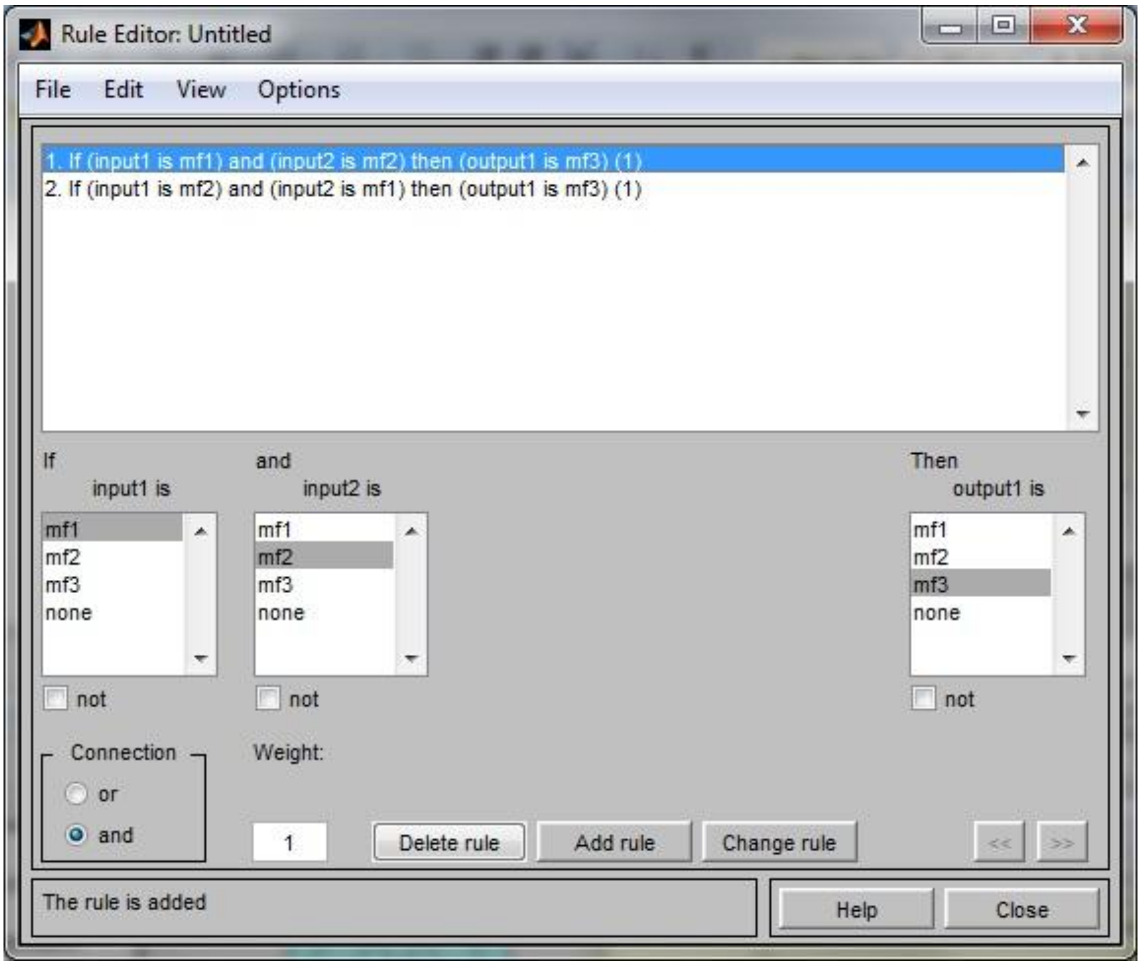
Type trimf

Params [-0.4 0 0.4]

Selected variable "input1"

Fig. 9. Membership Function Editor

Fuzzy rules are a set of rules that the fuzzy controller works according to. These rules can be determined in Rule Editor (figure 10).



1. If (input1 is mf1) and (input2 is mf2) then (output1 is mf3) (1)
2. If (input1 is mf2) and (input2 is mf1) then (output1 is mf3) (1)

Weight: 1

The rule is added

Fig. 10. Rule Editor

The other important tool in Fuzzy Logic Toolbox is the collection of Simulink blocks for fuzzy control systems. There are two different kinds of fuzzy controller blocks. One with ruleviewer and one without. Ruleviewer is a graphical tool that lets users see which rules are in effect in their system in real-time while running the system. Also blocks to determine the membership functions and their shapes are provided in the toolbox. The Simulink blocks can be seen in figures 11 and 12.

The Fuzzy Logic Controller blocks use FIS-files and they can be designed with FIS-editor also available in Fuzzy Logic Toolbox.

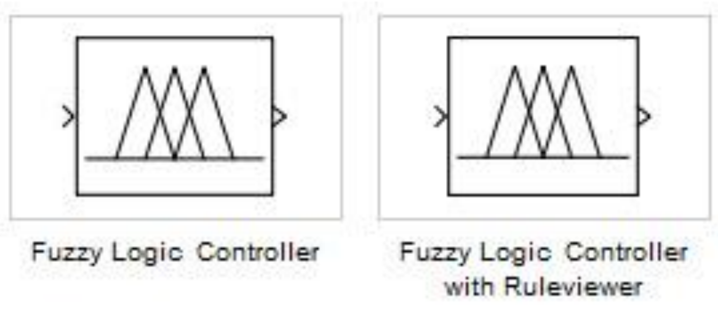


Fig. 11. Fuzzy Logic Controller Simulink blocks

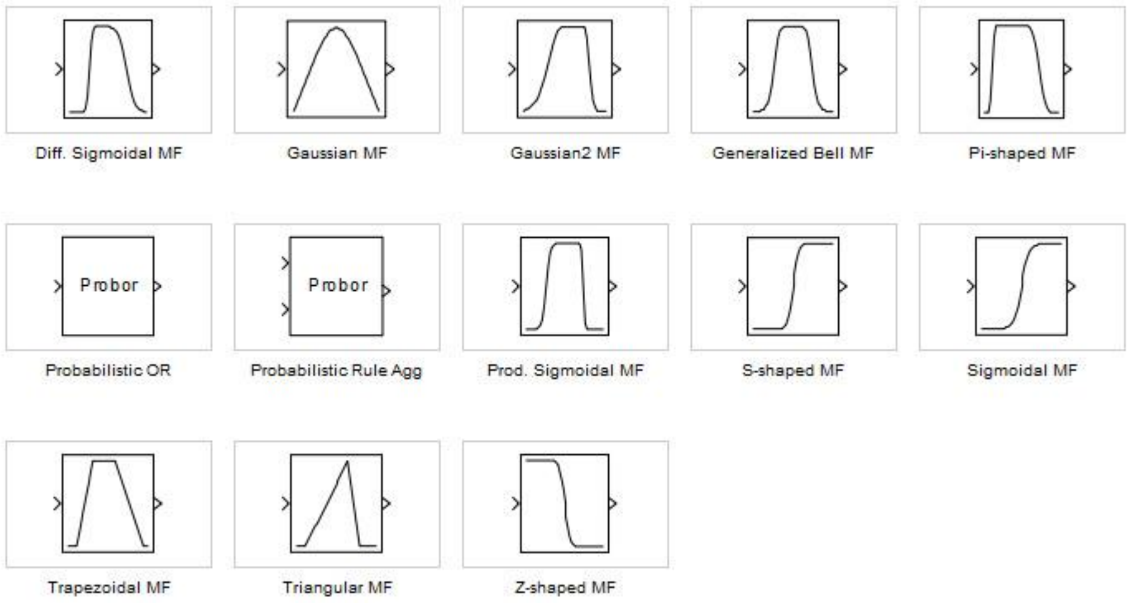


Fig. 12. Fuzzy Logic Membership Function Simulink blocks

2.6.3 Data Acquisition Toolbox

Data Acquisition Toolbox adds support for connecting a range of devices to MATLAB for data acquisition purposes. The connection can be used both ways: to and from MATLAB.

Also the toolbox adds six new blocks to Simulink library. The new I/O blocks are presented in figure 13.

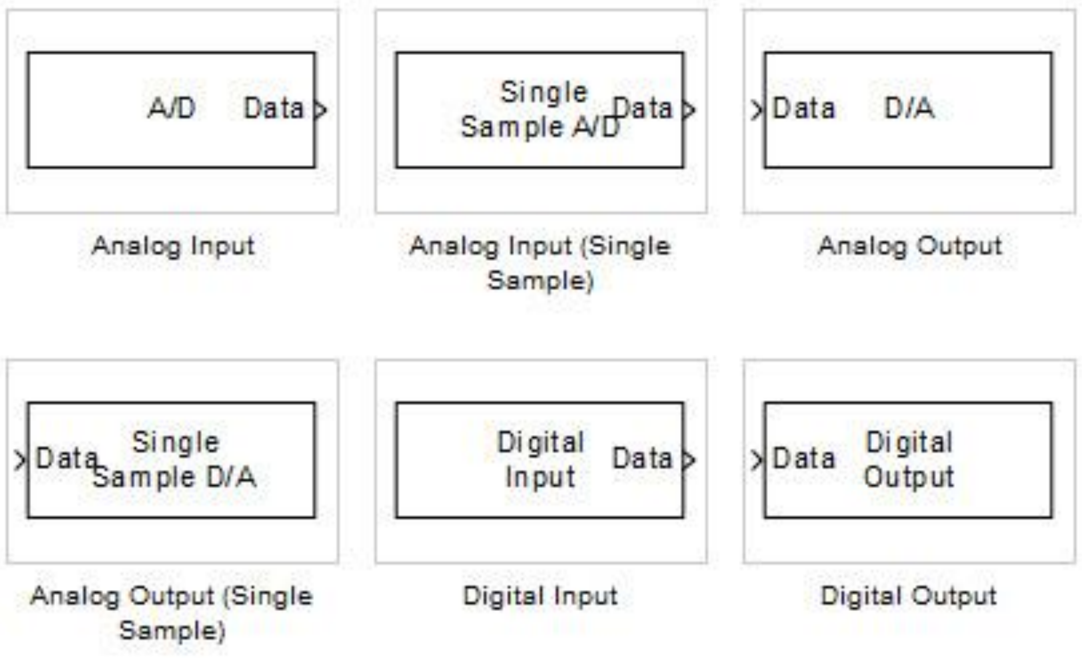


Fig. 13. Simulink blocks in Data Acquisition Toolbox

3 MODIFICATIONS TO WHEELCHAIR

The equipment used in this project consisted of three main devices: an electric-powered wheelchair, motion sensors and a computer. In addition to these devices various other peripherals were used in order to form control signals.

First the EPW needed to be able to receive input signals that are used for controlling the EPW. In normal case the signals are equivalent to movements of the joystick but since the project was to develop a hands-free system, the joystick control needed to be by-passed. The solution to produce the needed input signals in alternative way was to use a laptop running a Simulink based system that sends the input signals to the main controller of EPW through an I/O card and a screw terminal attached to the EPW. The layout of alternative input signal routes to the main controller is shown in figure 14.

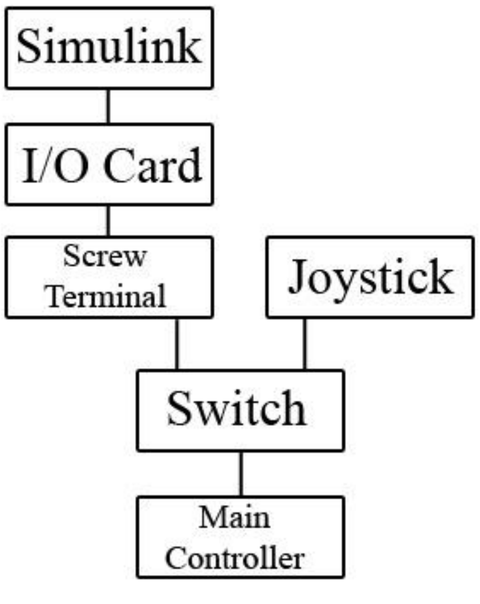


Fig. 14. Flowchart of data transfer in EPW control system

The modifications made included attaching a screw terminal on the side of the EPW and connecting a switch that allows choosing either joystick or hands-free control. The installed parts can be seen in figure 15.

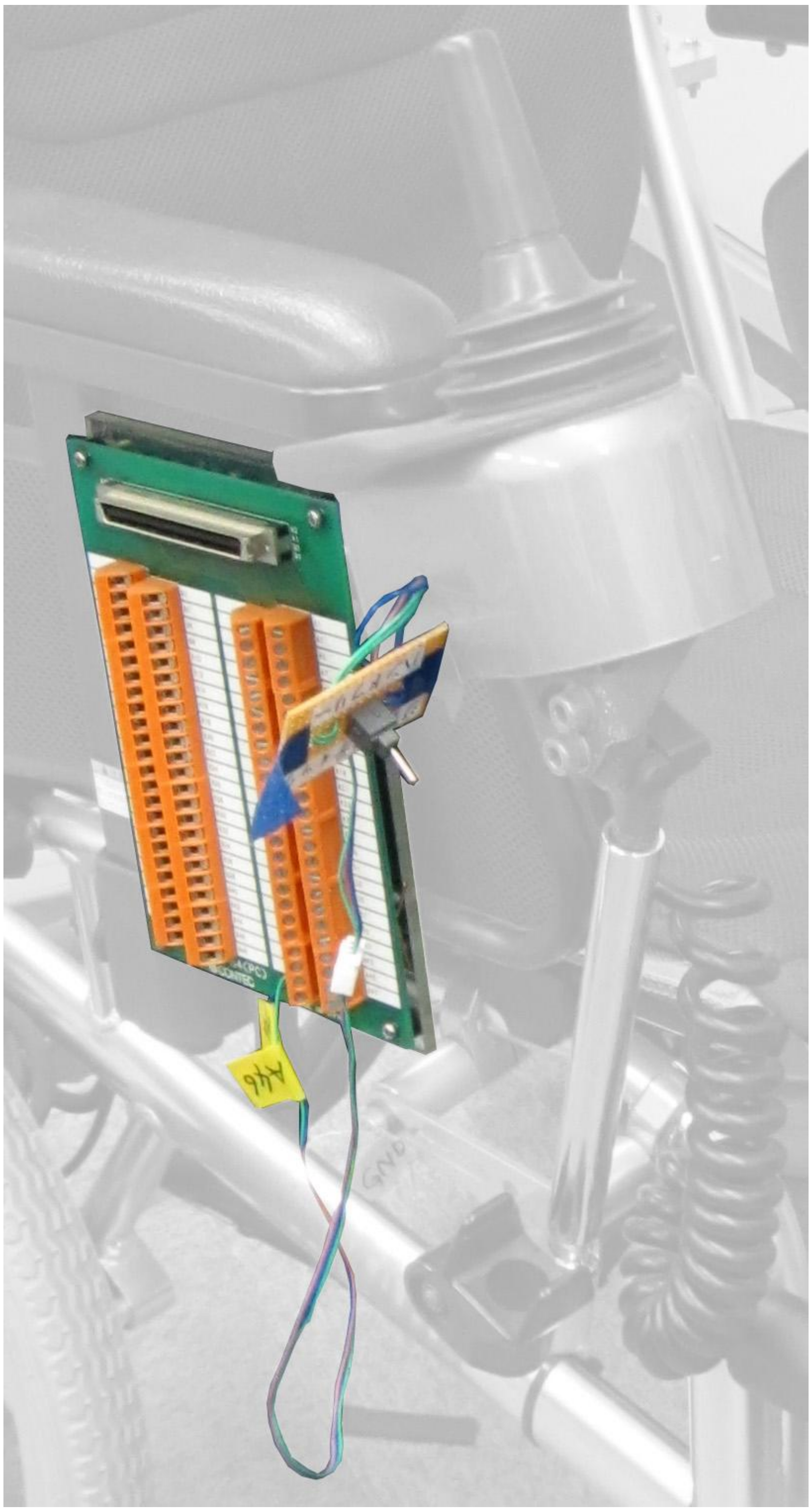


Fig. 15. Modifications made to EPW

3.1 ELECTRIC-POWERED WHEELCHAIR

The electric-powered wheelchair used in this project was the Towny Joy manufactured by Yamaha. The Towny Joy has a folding chassis. This makes it easy to store for transit. Drive system is a rear wheel drive with electric motors powering each rear wheel separately. Specifications of Towny Joy can be seen in appendix 2[12]. A picture of Yamaha Towny Joy EPW is shown in figure 16.

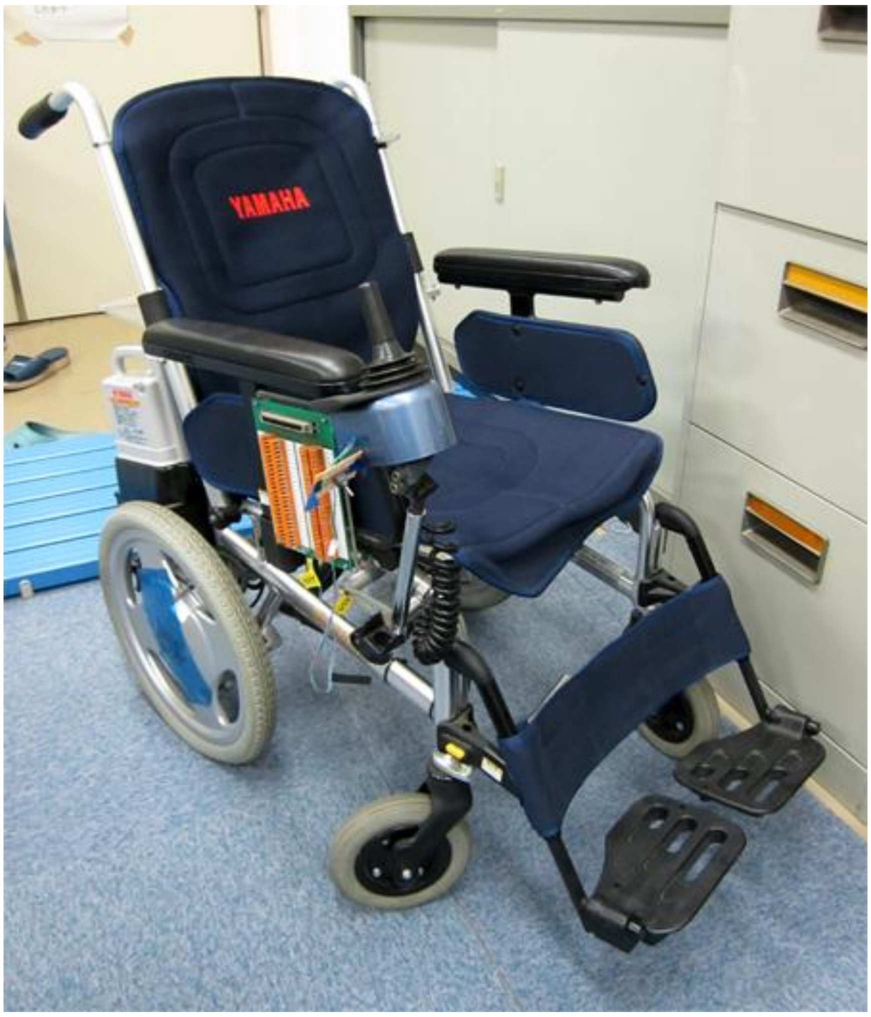


Fig. 16. Yamaha Towny Joy electric-powered wheelchair

A picture of one of the motors that powers the wheelchair can be seen in figure 17.

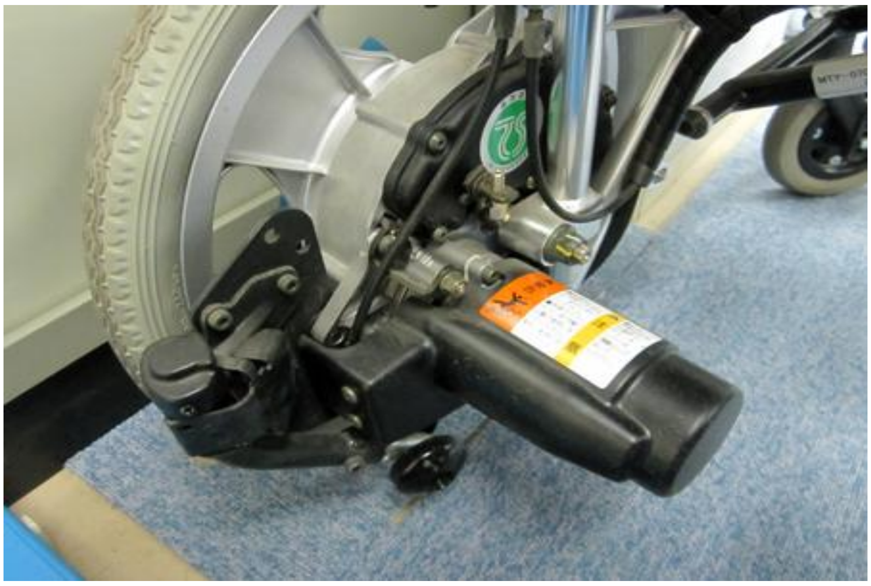


Fig. 17. One of the two motors powering the wheelchair

Yamaha Towny Joy uses a rechargeable NiMH dry cell battery. The battery is removable and comes with a separate charger unit. Output voltage is 24 V and its electric charge of 6.7 Ah is enough to power the wheelchair for 5 hours. The battery is protected by a 20 A automotive fuse. The battery of the EPW used is shown in figure 18.

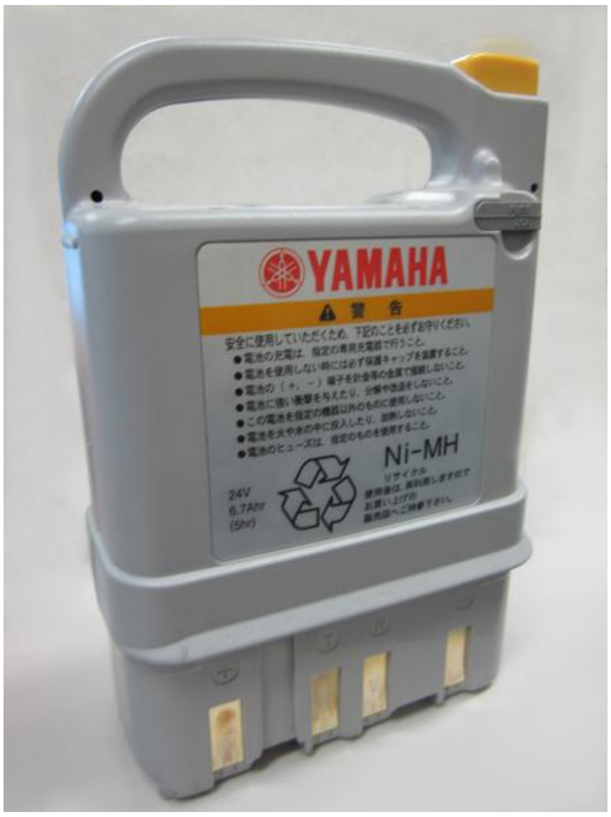


Fig. 18. The rechargeable NiMH battery used in Yamaha EPW

By default the Yamaha EPW uses a joystick (seen in figure 19) to control the wheelchair. In addition to the joystick there are also switches for power and sensitivity. For research purposes in this project the joystick controller was bypassed and the motion sensors were connected to a computer that was used for controlling the movements of the EPW.

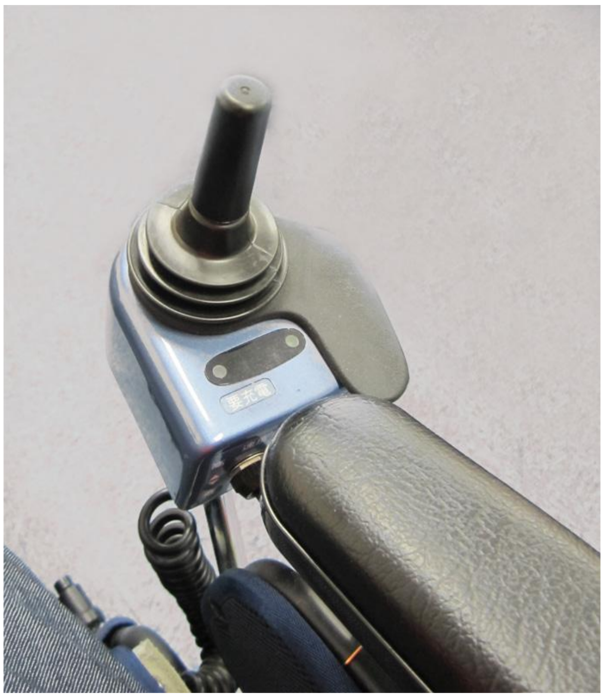


Fig. 19. A joystick that is used to control Yamaha EPW

The layout of the EPW control system is shown in figure 20.

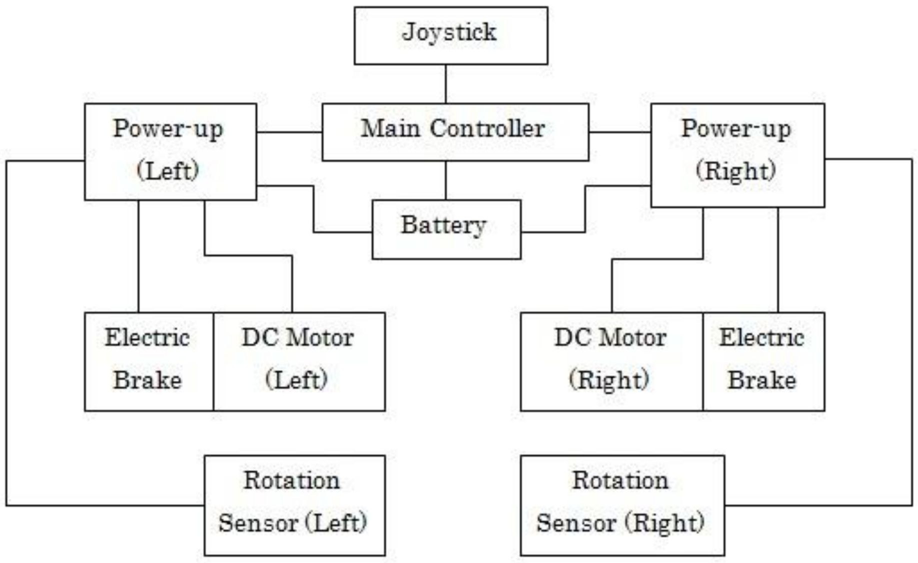


Fig. 20. Control System Layout of the Yamaha EPW

To determine what kind of signals are used for controlling the EPW some measurements were carried out. The goal was to measure the voltage the joystick sends to the main controller. The layout of possible signals based on the measurements done can be seen in figure 21.

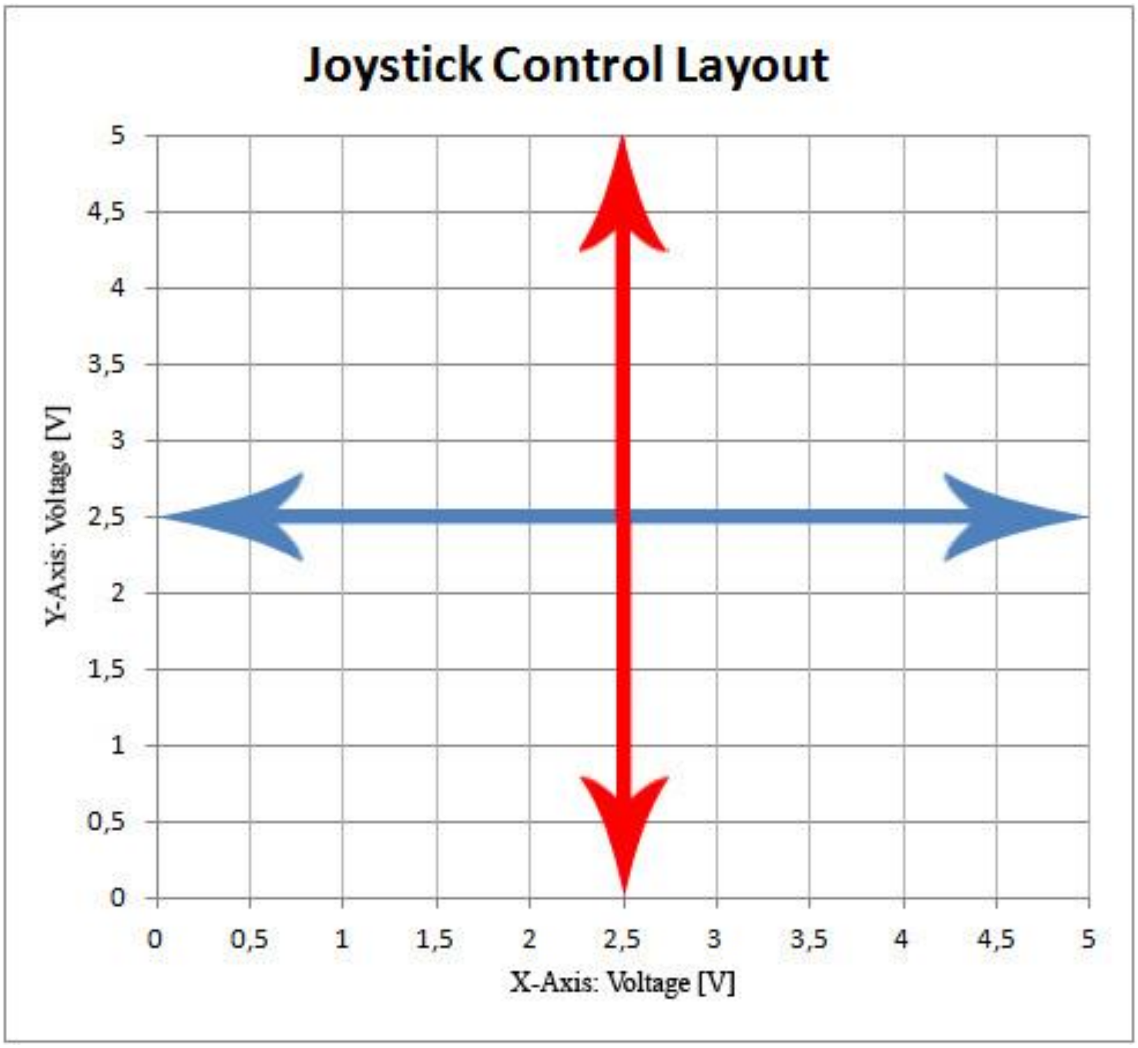


Fig. 21. Joystick Control chart of control signals

This chart can be interpreted that when X-Axis is 0 V, the wheelchair turns left. When the X-Axis is 5 V, the wheelchair turns right. Same principle is true for Y-Axis: 0 V means backwards and 5 V means forward. The wheelchair stays still when X and Y are both exactly 2.5 V. These X and Y-values will be used later on in this thesis.

3.2 MOTION SENSOR

IMU-Z by ZMP INC. is a small and lightweight motion sensor designed to include acceleration sensor, gyroscopic sensor and compass sensor in one unit. In other words this 9-axis sensor is composed of three individual sub-sensors that each is a 3-axis sensor on its own. One motion tracking system can consist of up to 28 units simultaneously. The more units are used simultaneously the more precise data of the movements can be gathered. IMU-Z can be connected to a computer through CAN-bus or wirelessly through Bluetooth. A picture of IMU-Z is shown in figure 22.

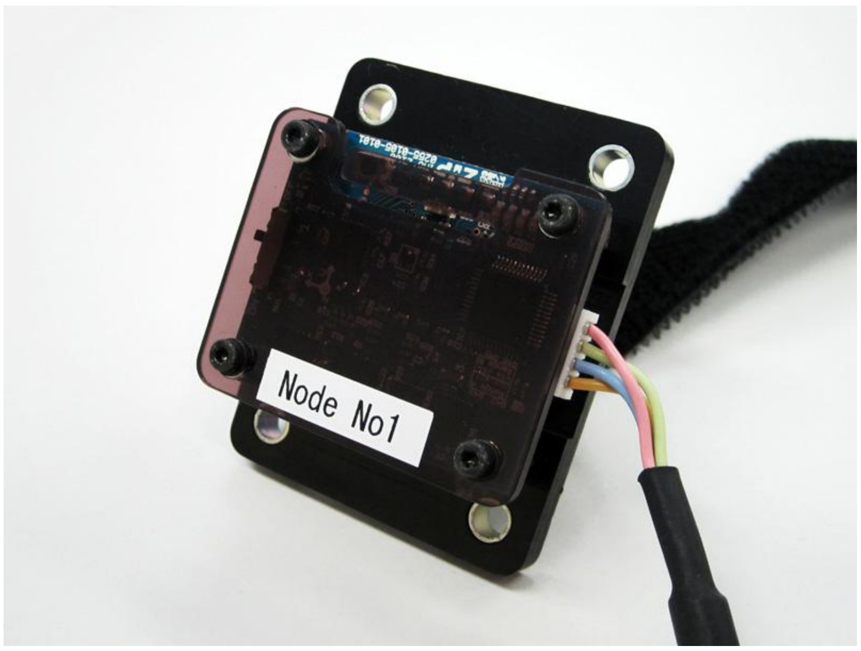


Fig. 22. ZMP IMU-Z motion sensor

Specifications of ZMP IMU-Z can be found in appendix 3[13]. For more info on CAN message formats used by ZMP IMU-Z please refer to appendix 4.

To make the IMU-Z system more mobile, a batterybox is available as an optional accessory. This makes it possible to use sensors without AC-adapter. The batterybox can be seen in figure 23.

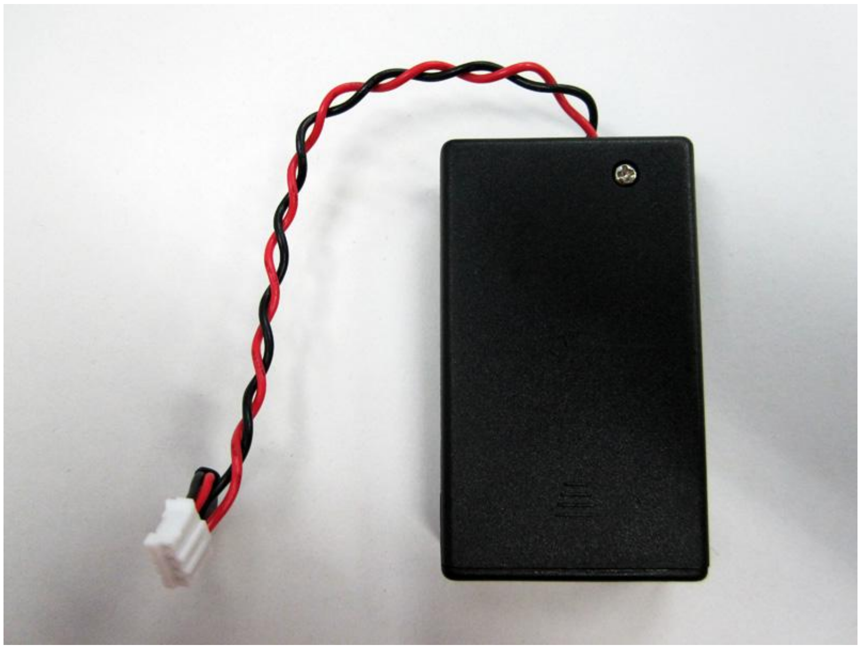


Fig. 23. Batterybox for IMU-Z

3.3 CAN TO USB ADAPTER

CANUSB is an CAN to USB adapter manufactured by Lawicel AB that is designed to give an affordable access to CAN devices through the USB port of a computer. When connected the adapter works like a standard serial RS-232 port. This means no special drivers are needed. On the other hand custom drivers can improve data transfer rates and allows bigger bus loads. Data transfers can be done in standard ASCII format. Figure 24 shows a picture of CANUSB adapter.

Fig. 24. Lawicel AB CANUSB adapter required for connection

Detailed specifications of CANUSB can be found in appendix 5[14].

The pin arrangement of the CANUSB adapter is presented in figure 25. As USB port provides the needed voltage there is no need to connect power to pin 9. Only pin 2 (CAN_L), pin 7 (CAN_H) and pin 3 (CAN_GND) are used.

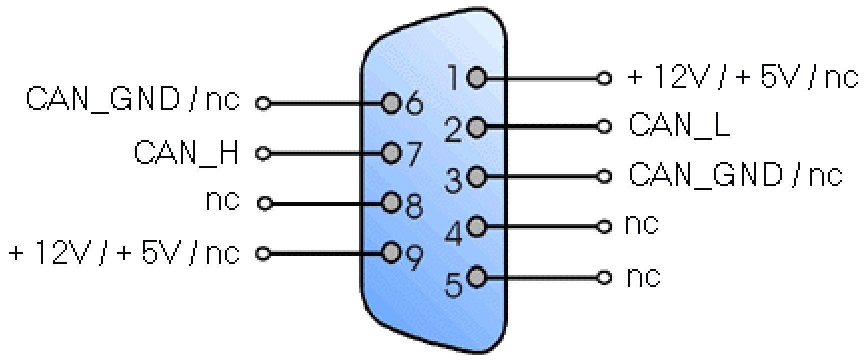


Fig. 25. Pin arrangement according to CiA recommendations DS102-1[14]

3.4 I/O PC CARD

ADA16-32/2(CB)F is a multi-function Type II size standard CardBus I/O card manufactured by CONTEC. It contains analog inputs (16-bit, 32 channels), analog outputs (16-bit, 2 channels), digital inputs (4 channels), digital outputs (4 channels) and counters (32-bit binary, 1 channel). A picture of the card is shown in figure 26.

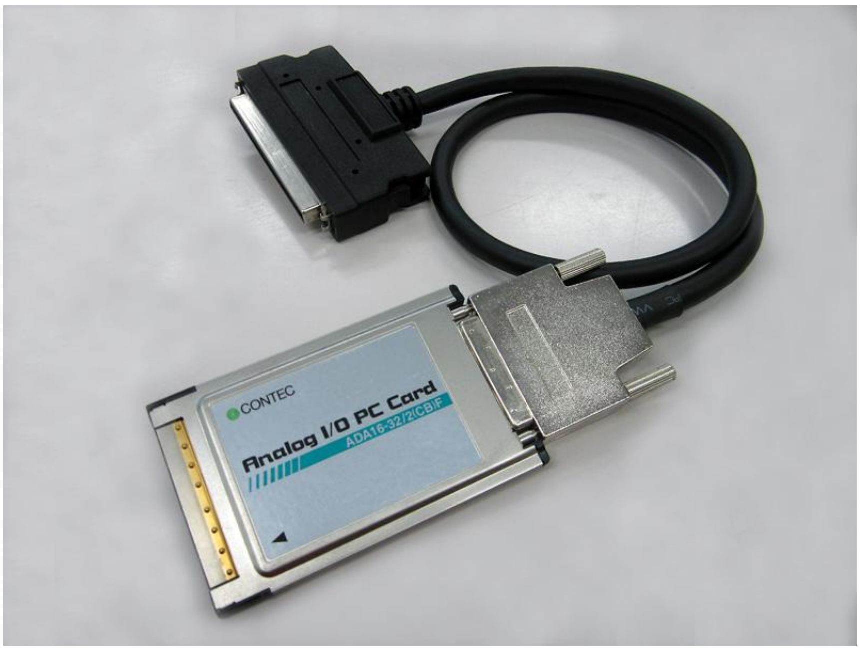


Fig. 26. CONTEC ADA16-32/2(CB)F Analog I/O PC Card

The device comes with driver libraries that support many programming platforms through Win32 API functions. Also various plug-ins are available that add support for MATLAB and LabVIEW softwares.

ADA16-32/2(CB)F comes included with an event controller for integrated hardware control. This can be used to build a simple yet effective PC-based measurement and control system that is ideal for controlling small-sized systems. The overview of the event controller can be seen in figure 27 below.

The arrows in the figure are the control signals between the event controller and I/O connections. The main control signals include clock signals and signals to start or stop operations.

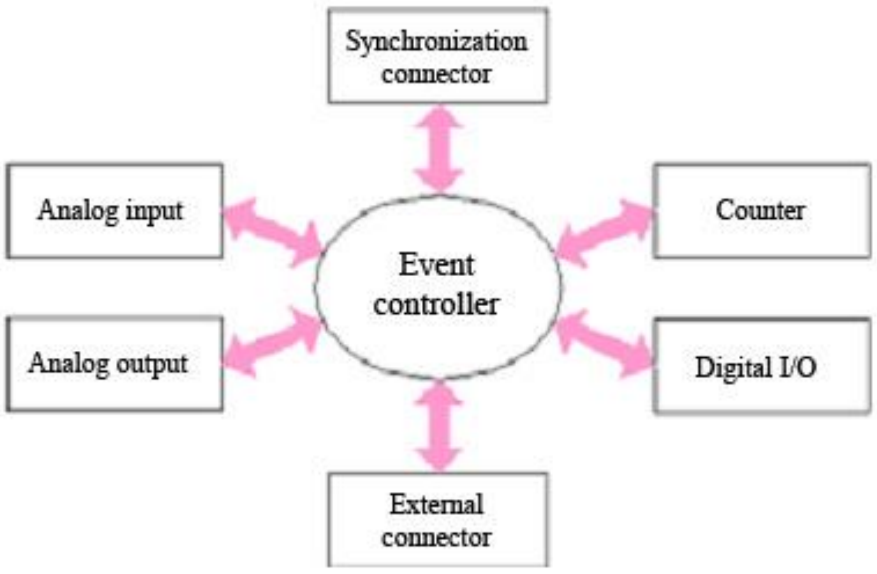


Fig. 27. Overview of event controller[15, p.1]

In order to use the I/O Card on MATLAB, installation of Data Acquisition library is needed. The file is available on CONTEC website.

3.5 SCREW TERMINAL

DTP-64(PC) is a screw terminal that relays the I/O wirings to a CONTEC I/O card through 96-pin half-pitch connector. An external device can be connected by fastening the wires using screws. The DTP-64(PC) with I/O wiring connected is presented in figure 28.

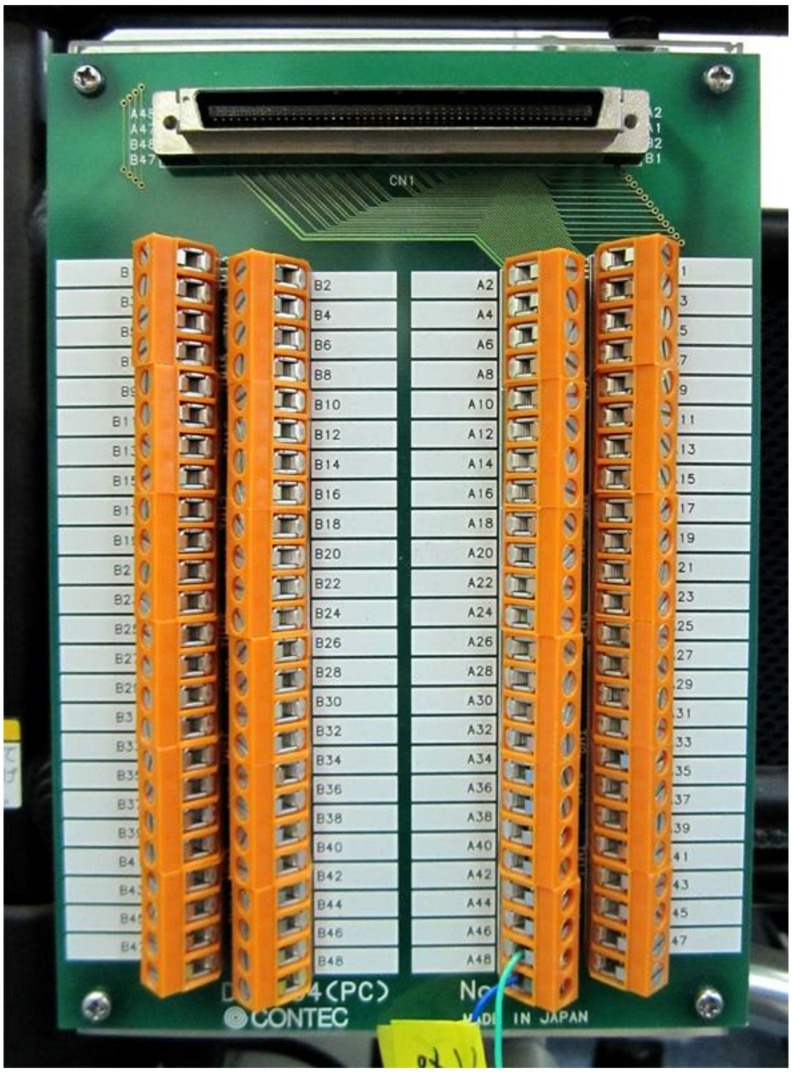


Fig. 28. CONTEC DTP-64(PC) Screw Terminal

The route of the control signal can be seen in the overview of DTP-64 connections (figure 29). The signal travels from left to right.

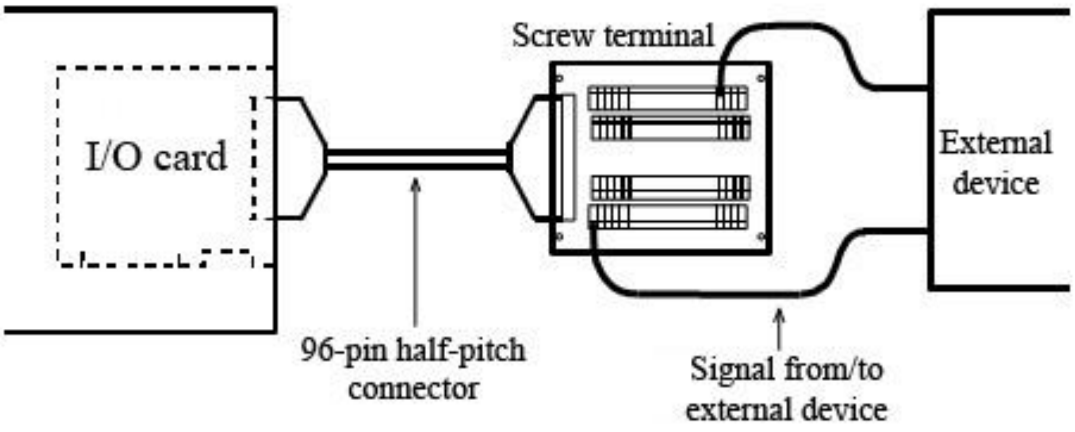


Fig. 29. Overview of CONTEC DTP-64(PC) Connections [16, p.2]

4 EXECUTION OF CONTROL SYSTEM

The next sections describe the execution of the control system for EPW. The main steps are configuration of devices and software, determination of the needed components in Simulink program and fitting the components together.

4.1 CONFIGURATION OF MOTION SENSOR

ZMP IMU-Z software installation CD contains everything that is needed for configuration of an IMU-Z unit. The installation program installs the drivers for IMU-Z and all needed software. For configuration IMU-Z Firmware Updater and IMU-Z Configuration Tool are needed.

The very first recommended thing to do before starting the actual configuration is to update the firmware the IMU-Z unit. Version 3 or higher works more reliably compared to previous versions that have had problems with saving settings. The firmware can be updated with IMU-Z Firmware Updater tool. ZMP IMU-Z Manual provides detailed instructions on firmware updates.

Parameters that can be changed and their units:

- period [ms]
- range for each sensor
    - acceleration [g]
        - 2, 4 or 8 g
    - gyroscope [deg/s]
        - 500 or 2000 deg/s
    - compass [G]
        - 0.7; 1.0; 1.5; 2.0; 3.2; 3.8 or 4.5 G
- node number (1-28)
- role
    - SingleBT
    - CAN-MasterBT
    - CAN-slave
- message format
    - text
    - binary

IMU-Z configuration is done with IMU-Z Configuration Tool program, pictured in figure 30.

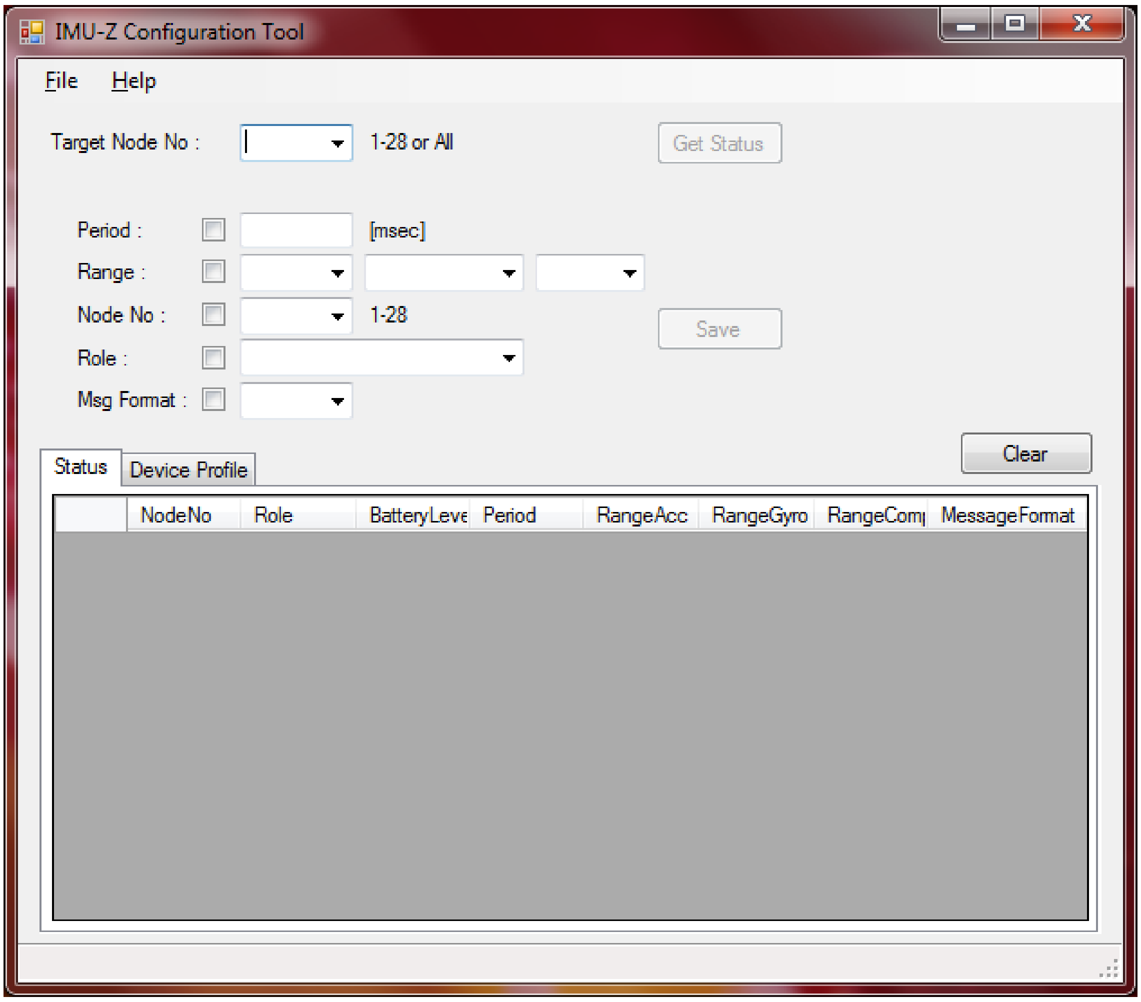


Fig. 30. IMU-Z Configuration Tool

Configuration is started by opening the connection to the device. This can be done from File->Open. Next a new window appears (figure 31).

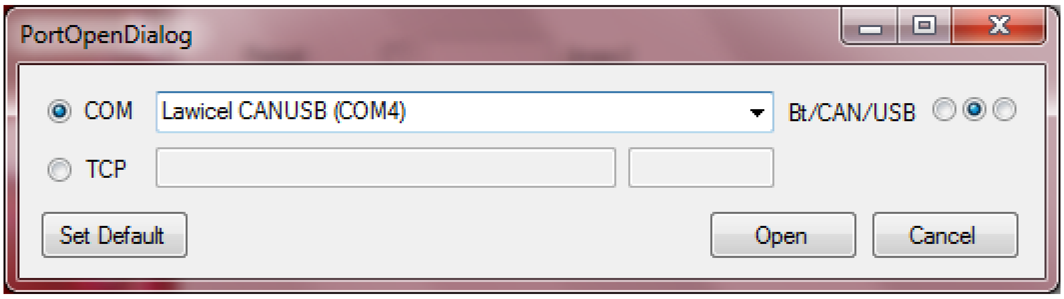
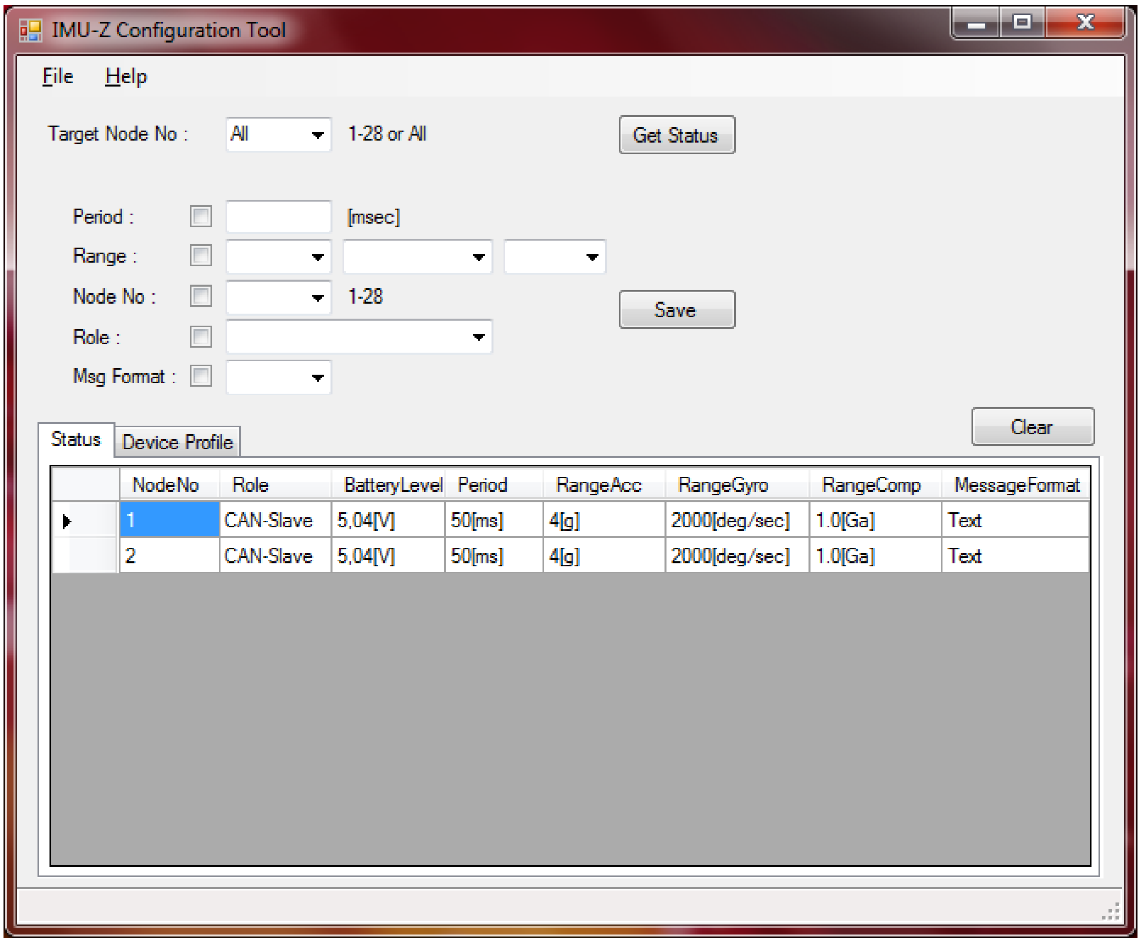


Fig. 31. IMU-Z Configuration: Connection type

COM-interface and a connection type need to be chosen. IMU-Z supports Bluetooth (Bt) and CAN standards. In this case CAN was used. The connection is opened by clicking “Open”. The window closes and configuration continues in the main window.

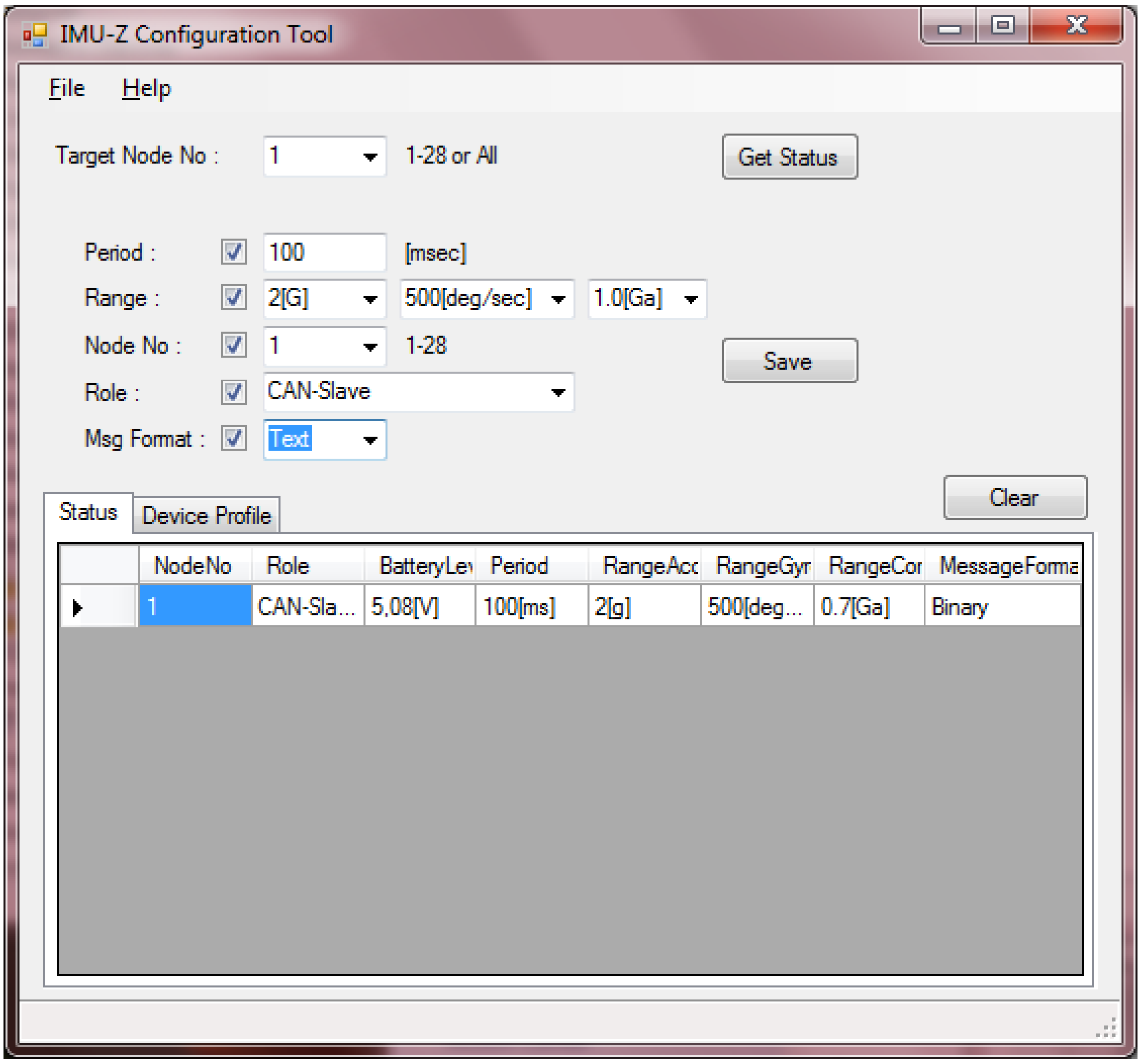
Button “Get Status” shows the current saved parameters. An example of a two sensor system status is presented in figure 32.



NodeNo	Role	BatteryLevel	Period	RangeAcc	RangeGyro	RangeComp	MessageFormat
1	CAN-Slave	5,04[V]	50[ms]	4[g]	2000[deg/sec]	1.0[Ga]	Text
2	CAN-Slave	5,04[V]	50[ms]	4[g]	2000[deg/sec]	1.0[Ga]	Text

Fig. 32. IMU-Z Configuration: Get Status

The parameters were changed and saved by clicking “Save”. The parameters used in this configuration are shown in figure 33. The parameters were chosen by testing all different options and trying to achieve as fast output response time as possible when the IMU-Z was moved in significant way. Each project are different therefore the desired parameter values need to be determined for each system separately.



Target Node No : 1 1-28 or All

Period : ✓ 100 [msec]

Range : ✓ 2[G] 500[deg/sec] 1.0[Ga]

Node No : ✓ 1 1-28

Role : ✓ CAN-Slave

Msg Format : ✓ Text

NodeNo	Role	BatteryLev	Period	RangeAcc	RangeGyr	RangeCor	MessageForma
1	CAN-Sla...	5,08[V]	100[ms]	2[g]	500[deg...	0.7[Ga]	Binary

Fig. 33. IMU-Z sensor parameters

4.2 DATA ACQUISITION IN MATLAB

CONTEC I/O cards require a dedicated library called ML-DAQ in order to communicate with MATLAB. The library is available for download at CONTEC website. The library file (mwcontec.dll) is to be extracted to following location:

- $MATLAB/toolbox/daq/daq/private
- $MATLAB is the directory where the MATLAB is installed.

The registration of the CONTEC I/O card is done by execution of the following commands in Command Window in MATLAB:

- rehash toolboxcache
- daqregister(‘contec’)

Confirmation of valid device registration can be seen by execution of the following command in Command Window:

- daqhwinfo('contec')

Registration of CONTEC I/O card in MATLAB enables the use of Simulink blocks in Data Acquisition Toolbox. The configuration of analog output block used in this system is introduced later on.[17, p.4-5]

4.3 ZMP IMU-Z MATLAB PACKAGE

The most important piece in the Simulink program is the IMU-Z MATLAB package from ZMP. It contains the MATLAB libraries needed to read data from IMU-Z sensors in MATLAB environment. The package is a commercial product that is available as common license (20,000 yen) or academic license (10,000 yen).

The package contains the libraries and an example program files. The example program contains Simulink program for reading the data from IMU-Z sensor and M-files that set up the communication through the libraries. Also a graphical user interface (GUI) code in M-file is included. The program can be started by running the M-file "IMUZ_START.m". The code in IMUZ_START.m can be found in appendix 6.

The example Simulink program can be seen in figure 34.

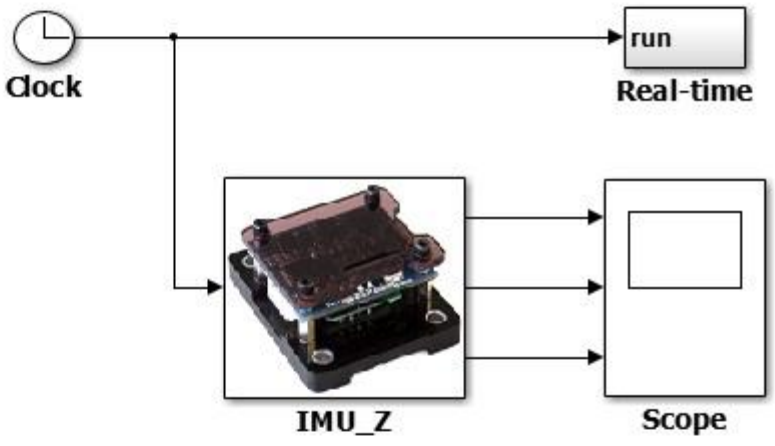


Fig. 34. Example Simulink program

As seen above the example program is very simple. The purpose of the program is to read measurement data from the IMU-Z sensor in real-time. The “Real-time” block contains a subsystem that synchronises the Simulink clock with real time. The “Real-time” subsystem is presented in figure 35.

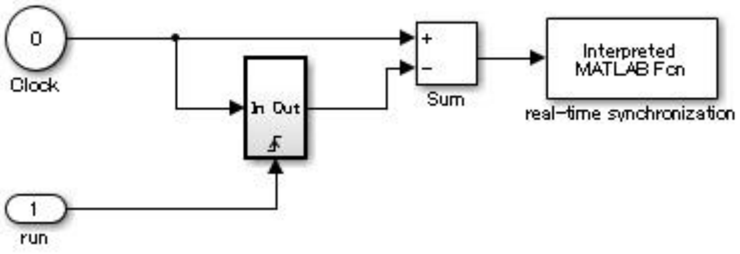
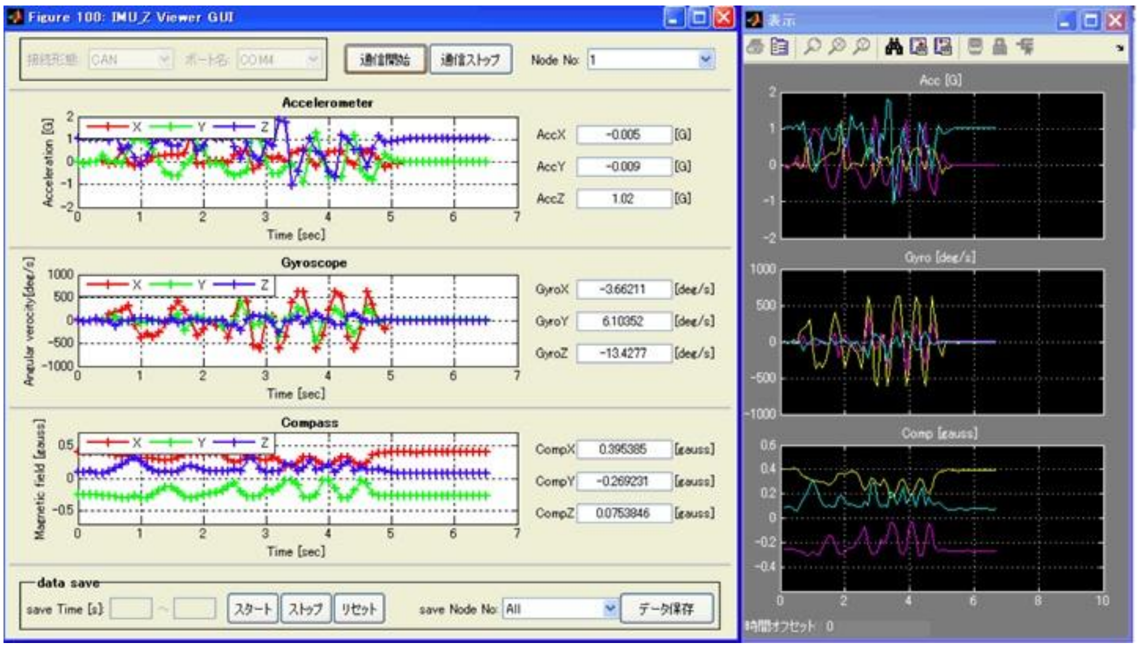


Fig. 35. A subsystem for synchronisation of Simulink clock with real time

The last block of this subsystem contains functions located in M-file ”waitfortreal.m”. The code can be found in appendix 7.

The ”IMU_Z” block contains functions from M-file “sfun_Imuz_sp.m” for loading libraries, reading measurement data and plotting functions. The code can be found in appendix 8. Output signals from this block are acceleration, gyro and compass data. Each of these contains information of said signal in three dimensions.

Figure 36 presents an example of the user interface and measurement data.



Node No: 1

AccX -0.005 [G]
AccY -0.009 [G]
AccZ 1.02 [G]

GyroX -3.66211 [deg/s]
GyroY 6.10352 [deg/s]
GyroZ -13.4277 [deg/s]

CompX 0.395385 [gauss]
CompY -0.269231 [gauss]
CompZ 0.0753846 [gauss]

save Time [s]: ~
save Node No: All

Fig. 36. 9-axis measurement data in user interface and oscilloscope[18]

This example program was used as a platform to build on when the control system for EPW was developed. The M-files work as they are and there was no need to edit them for this project.

For ease of use the user interface was translated from the original Japanese language to English. The translated version is shown in figure 37.

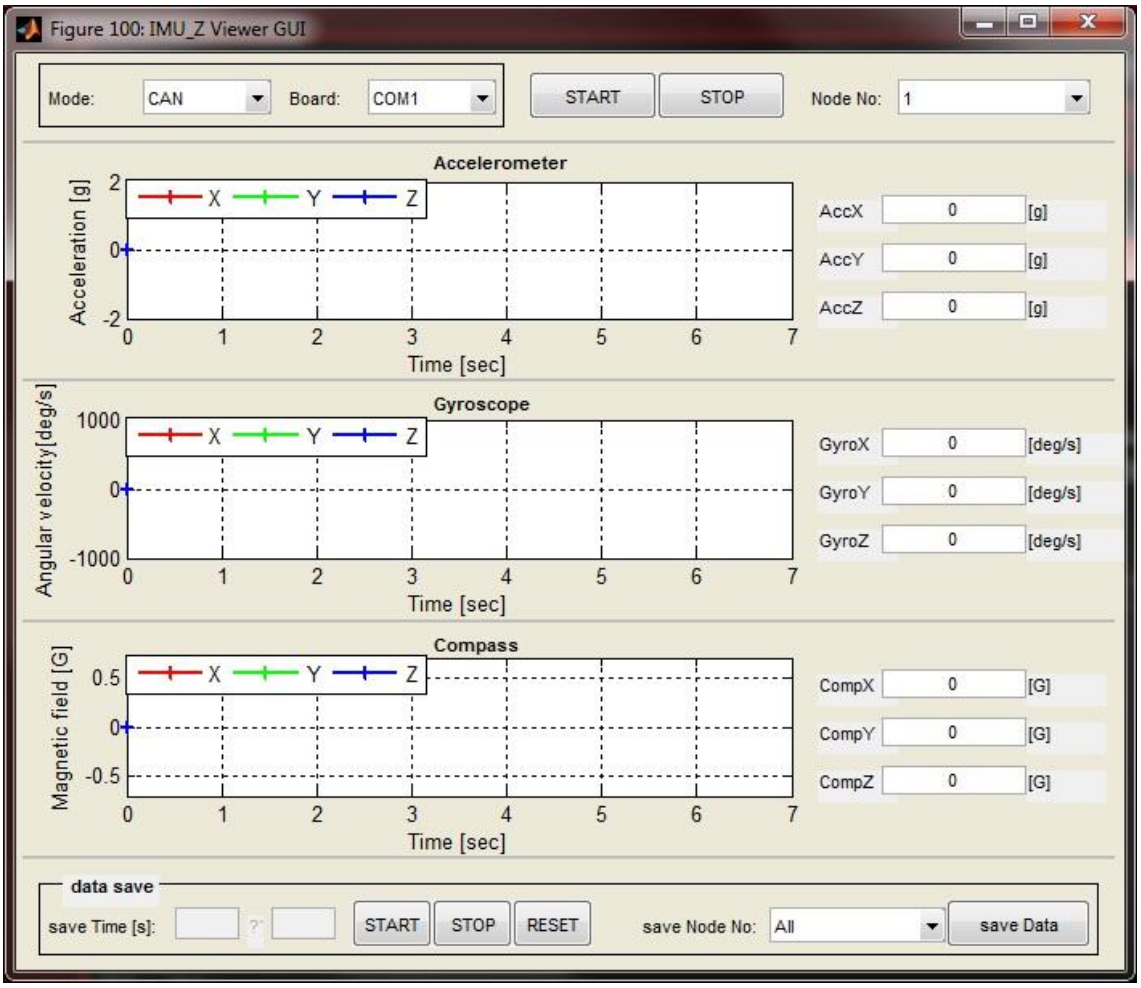


Fig. 37. Translated version of the user interface

The user interface can be used for starting and stopping the measurements. There is also a possibility to save measurement data. Button “START” near the top of the window runs the M-file “Imuz_CAN_start.m” (appendix 9). Button “STOP” near the top of the window runs the M-file “Imuz_CAN_stop.m” (appendix 10). Button “save Data” runs the M-file “Imuz_data_save.m” (appendix 11).

Communication mode and port can be chosen from drop-down lists. Because of this there is no need to change the communication parameters in M-files. All configurations can be done graphically in GUI and Simulink model.

4.4 PROCESSING OF MEASUREMENT DATA

Using the example Simulink model the measurement data can be read from the IMU-Z sensor with the ”IMU_Z” block. The three output signals coming out of the block are acceleration, gyroscope and compass data. Each one contains 3-axis data.

As mentioned before in section 3.1 the control signals that ultimately goes to the main controller of the EPW are in range of 0...5 V for X and Y-axis each. Upon closer inspection of the wirings of the joystick it was noticed that there are separate wires for X and Y-axis movement. For this reason also two output channels were needed in Simulink program. Now it is known how many signals are needed and what range they need to be in.

For the purpose of processing the data into format that could be used, a pre-made Simulink model was used. It was originally made for another motion sensor but it was modified to work with IMU-Z. In the Simulink model that was used in this thesis for data processing, only acceleration and gyroscope data was used. The model can be seen in figure 38.

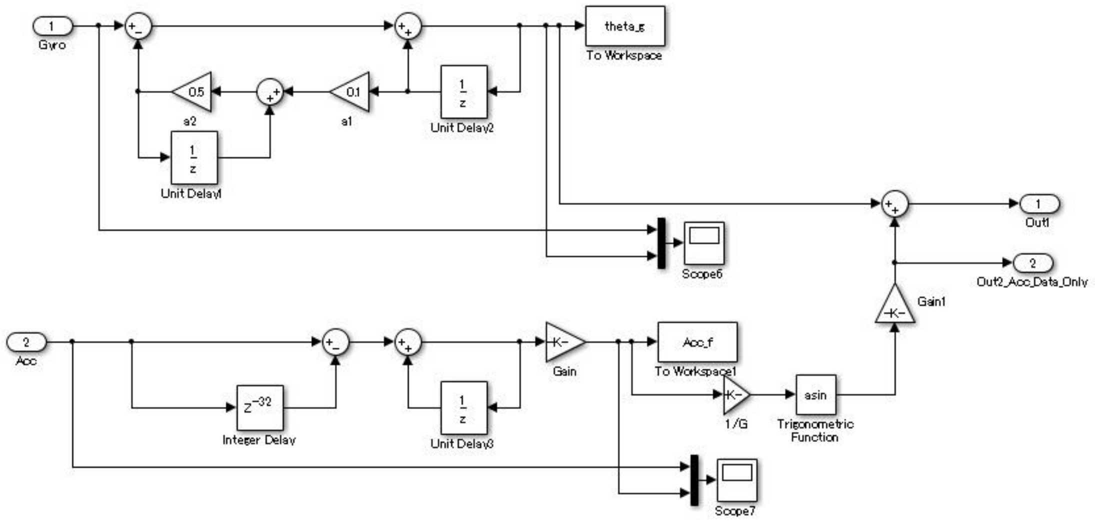


Fig. 38. Processing of acceleration and gyroscope data

The top part of the model is for processing the data from gyroscope and the bottom part processes the acceleration data. From these two the model calculates the angle of the IMU-Z sensor. This processed info can be used in the next step of the program.

4.5 FUZZY LOGIC CONTROLLER

Using fuzzy logic control system for projects like this thesis is a very convenient way to create a human like control scheme. It is a very straightforward system to be designed and suits a vast range of applications in the field of engineering.

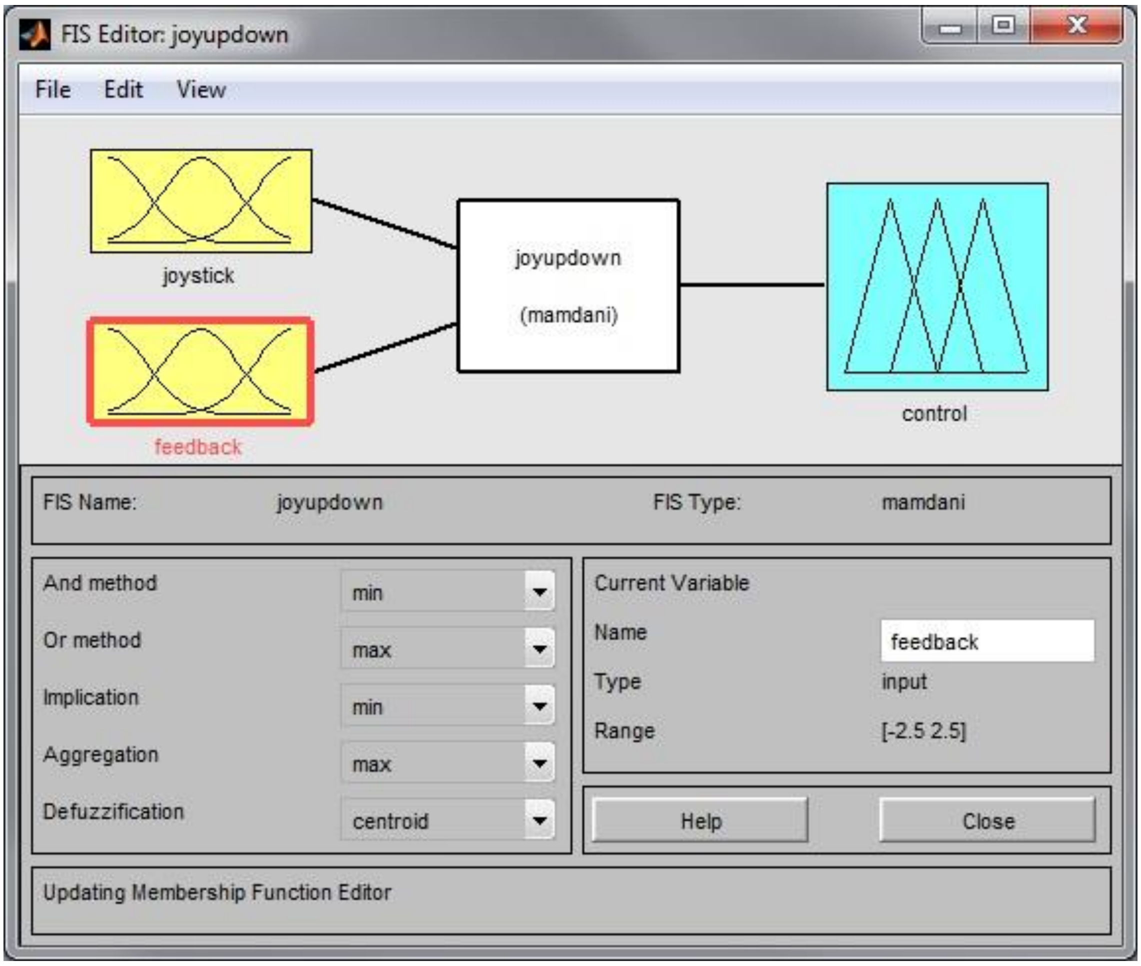
Fuzzy logic controller allows a stepless speed variation for the motors of the EPW instead of a predetermined cases that only allow a certain number of different speed variations. Also fuzzy logic can be used for scaling the signals in order to have them in range that the application required them to be.

The control system requires two separate fuzzy logic controller blocks because the signals that simulate the movement of joystick need to be kept separated in X and Y-axis.

Each fuzzy logic controller block uses two input signals: processed measurement data and a feedback connection was used to give a smoother control. The output signal will be in range of -2.5...+2.5 V.

The operation of the fuzzy logic control has been explained in next six figures (fig. 39-43).

Figure 39 shows the overview of the fuzzy logic controller that controls the forward and backward movement of the EPW. Two input signals called ”joystick” and ”feedback” will determine the output signal called ”control” according to the set of fuzzy rules.



FIS Name: joyupdown

FIS Type: mamdani

And method: min

Or method: max

Implication: min

Aggregation: max

Defuzzification: centroid

Current Variable

Name: feedback

Type: input

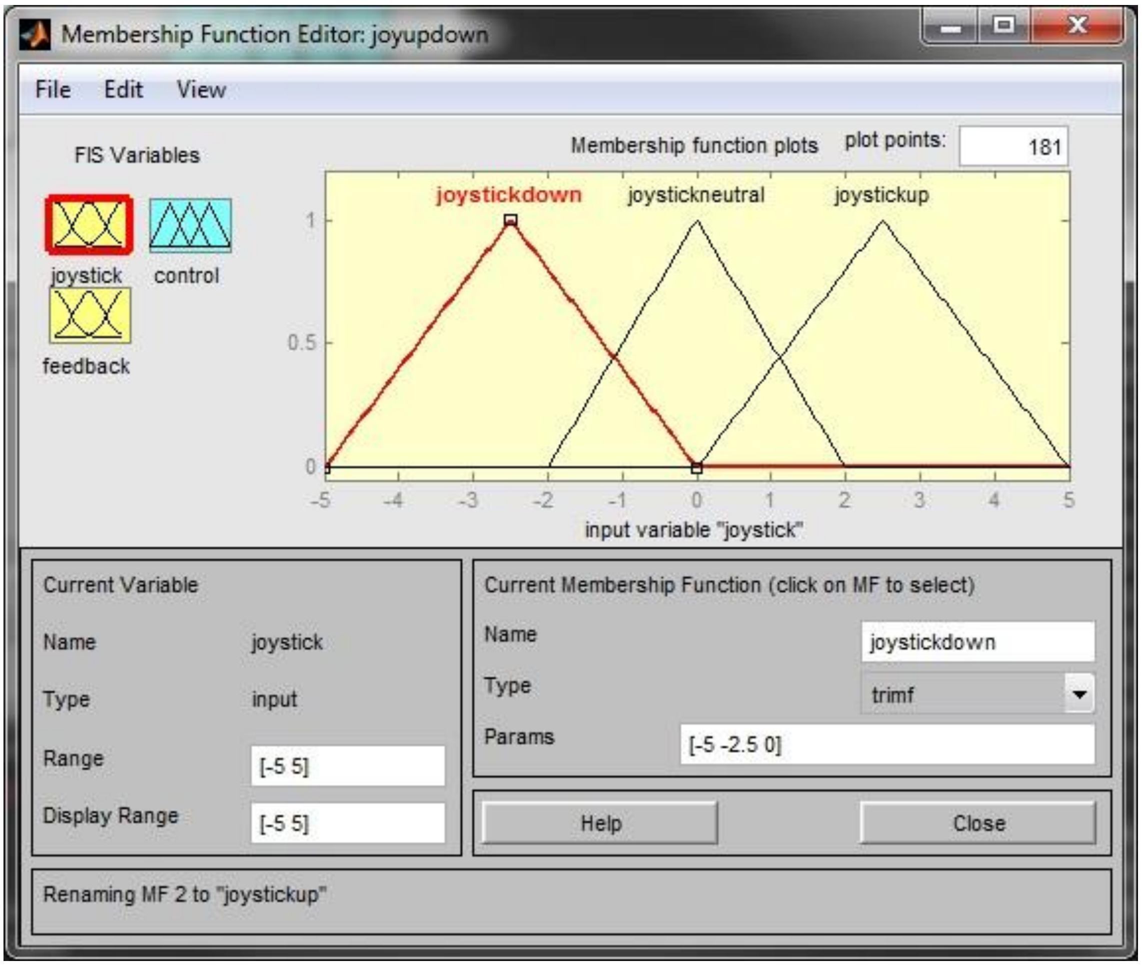
Range: [-2.5 2.5]

Updating Membership Function Editor

Fig. 39. Overview of fuzzy logic controller of forward and backward movement

In figure 40 a closer look at ”joystick” signal is taken. This signal will be the more dominant one that the output signal is based on. There are three different membership functions defined.

Function membership ”joystickdown” is ranged -5...0 V and its peak is at -2.5 V. This works so that also signals of -2.5 V and below are considered as part of this membership. Respectively ”joystickup” is ranged 0...+5 V and peaks at +2.5 V. Between these two function memberships is ”joystickneutral” that represents the situation when IMU-Z unit is in neutral position.


Membership Function Editor: joyupdown

File Edit View

FIS Variables

Membership function plots plot points: 181

Current Variable

Name joystick

Type input

Range [-5 5]

Display Range [-5 5]

Current Membership Function (click on MF to select)

Name joystickdown

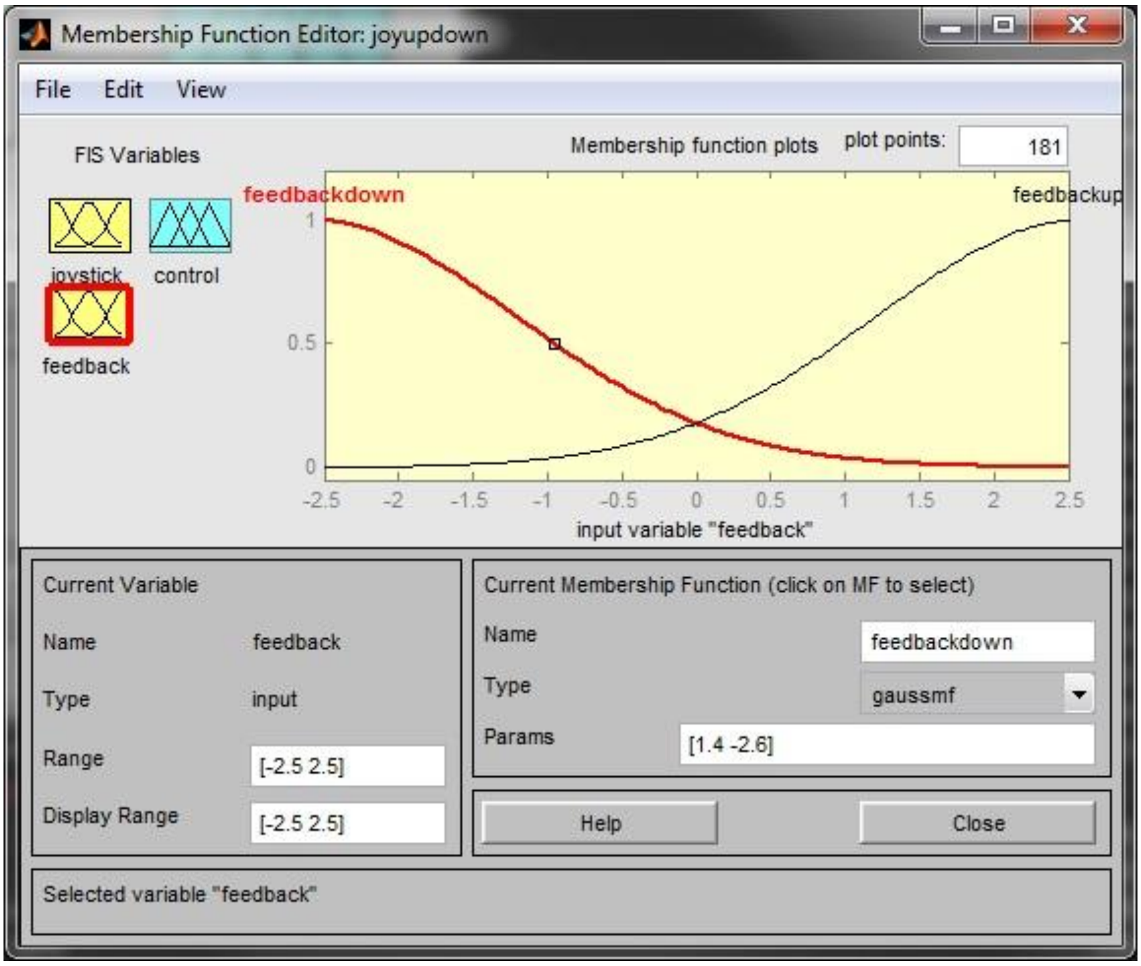
Type trimf

Params [-5 -2.5 0]

Renaming MF 2 to "joystickup"

Fig. 40. Membership functions of fuzzy logic controller of forward and backward movement

Input signal ”feedback” is a feedback signal that tells the fuzzy controller the current direction of movement. This is used so that the EPW would not do any sudden changes in movement. For example if the EPW is moving backwards but the IMU-Z gives a command to go forward, the fuzzy controller would stop the movement first smoothly and after that the forward movement can be commenced. Fuzzy set for signal “feedback” can be seen in figure 41.


Membership Function Editor: joyupdown

Current Variable

Name feedback

Type input

Range [-2.5 2.5]

Display Range [-2.5 2.5]

Current Membership Function (click on MF to select)

Name feedbackdown

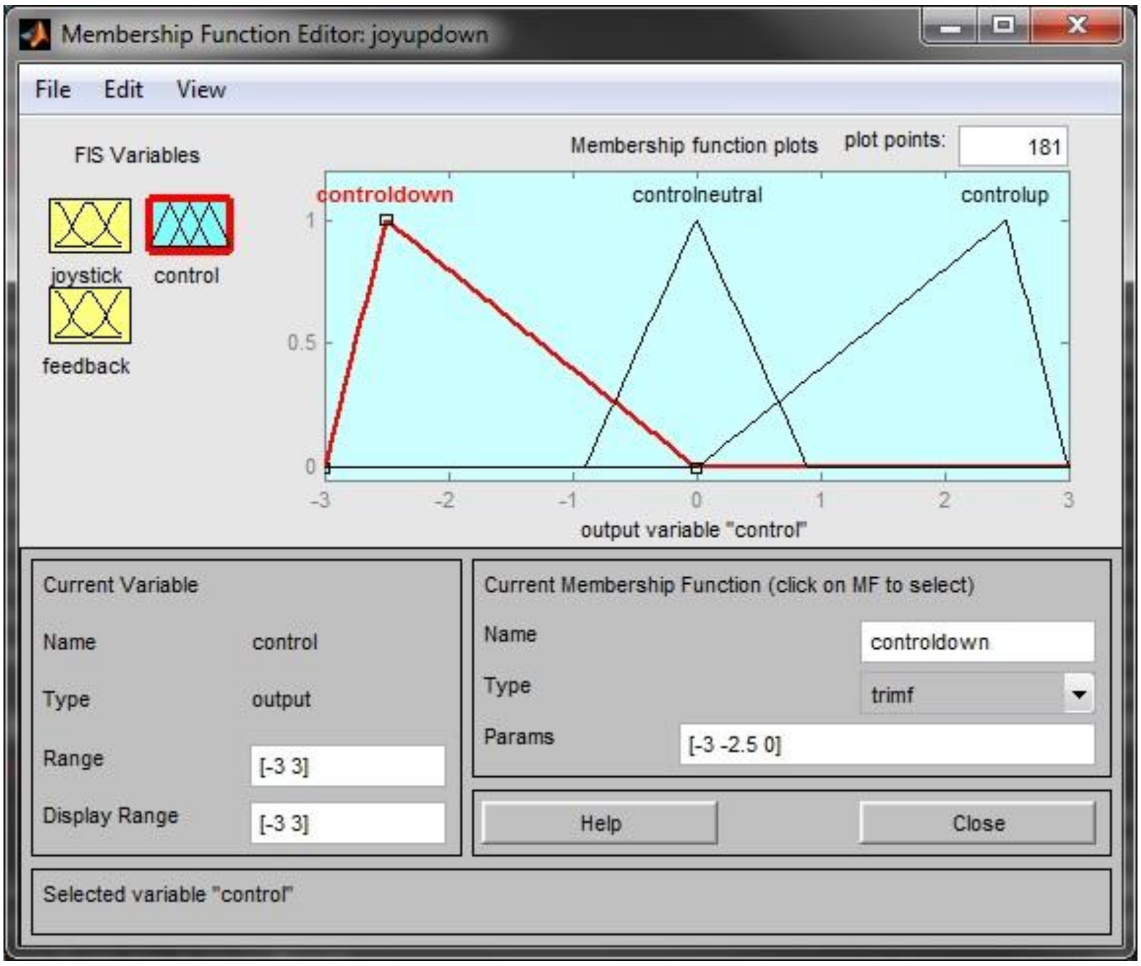
Type gaussmf

Params [1.4 -2.6]

Selected variable "feedback"

Fig. 41. Membership functions of feedback signal of fuzzy logic controller of forward and backward movement

Figure 42 shows the output signal ”control”. The membership functions shown correspond directly to what signal is sent to the EPW through I/O card.



Current Variable

Name: control

Type: output

Range: [-3 3]

Display Range: [-3 3]

Current Membership Function (click on MF to select)

Name: controldown

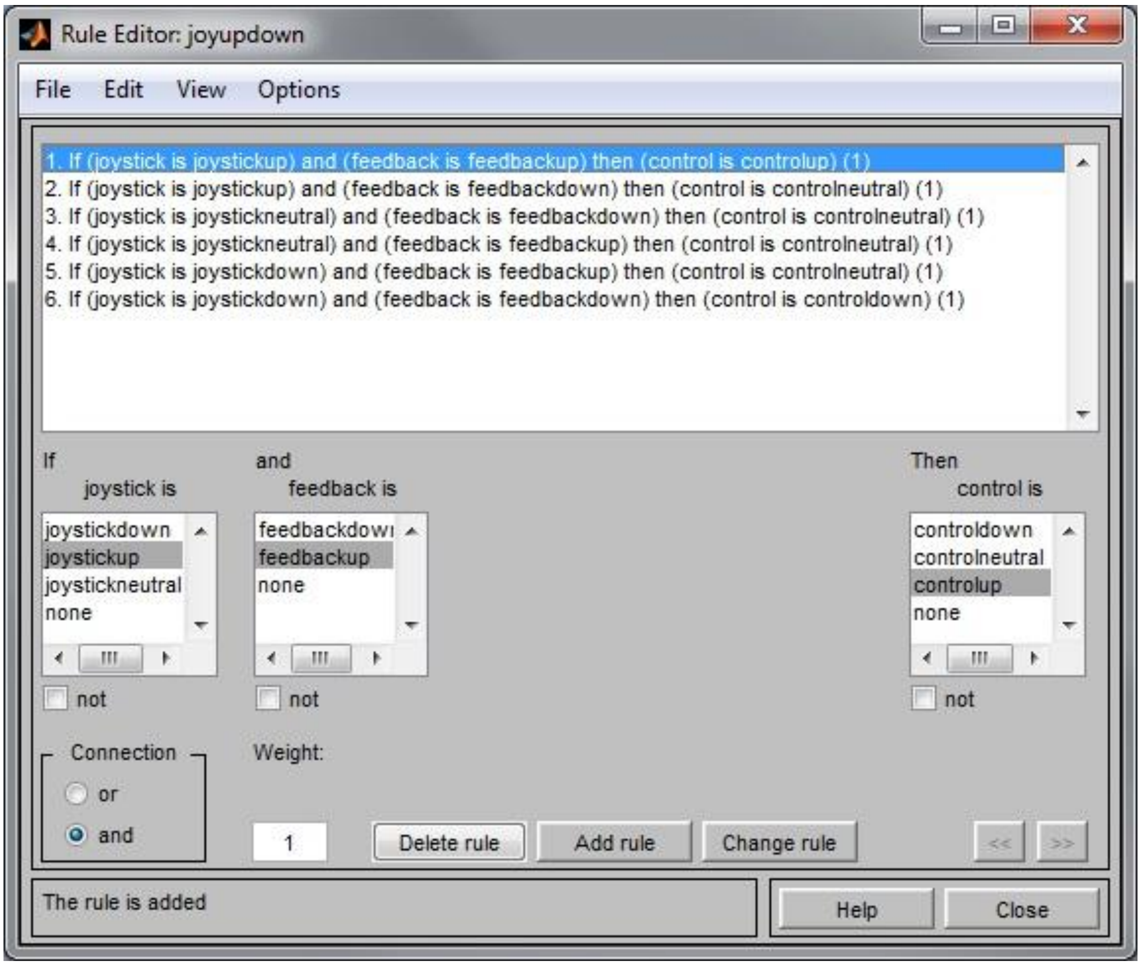
Type: trimf

Params: [-3 -2.5 0]

Selected variable "control"

Fig. 42. Membership functions of output signal of fuzzy logic controller of forward and backward movement

The set of fuzzy rules are presented in figure 43. These rules are very human like as they follow a simple formula. The formula is as following: if input A is X and input B is Y then output is Z.


1. If (joystick is joystickup) and (feedback is feedbackup) then (control is controlup) (1)
2. If (joystick is joystickup) and (feedback is feedbackdown) then (control is controlneutral) (1)
3. If (joystick is joystickneutral) and (feedback is feedbackdown) then (control is controlneutral) (1)
4. If (joystick is joystickneutral) and (feedback is feedbackup) then (control is controlneutral) (1)
5. If (joystick is joystickdown) and (feedback is feedbackup) then (control is controlneutral) (1)
6. If (joystick is joystickdown) and (feedback is feedbackdown) then (control is controldown) (1)

If joystick is: joystickup
and feedback is: feedbackup
Then control is: controlup
Connection: or [✓ and]
Weight: 1
The rule is added

Fig. 43. Fuzzy rules for fuzzy logic controller of forward and backward movement

After setting the parameters for the fuzzy controller, it can be saved as an FIS-file. This file can be also created in text editor instead of the graphical editor. The contents of the FIS-file can be seen in appendix 12.

The fuzzy logic controller for turning the EPW is almost identical to the controller for forward and backward movement. Therefore it will be presented in appendix 13.

4.6 ANALOG OUTPUT

An analog output block was needed to feed the already processed signals into the EPW. Therefore an analog output block was placed into the Simulink model. The used block with its parameters can be seen in figure 44.

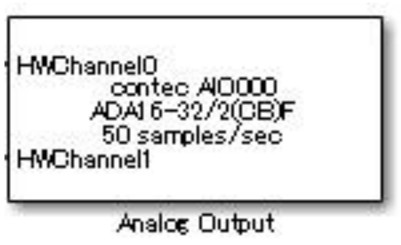


Fig. 44. Analog Output block

The analog output block has two channels. HWChannel0 is for turning the EPW and HWChannel1 is for going forward and backwards. 50 samples per sec was found to be a sufficient rate of sampling.

4.7 BUILDING A CONTROL SYSTEM

The goal for the developed Simulink program was to get input data to the control system from the IMU-Z sensor, process it into two signals that can be transferred to the EPW using the I/O card.

The previously presented main components are not all that is needed to build a complete control system. A number of additional blocks are needed to fill the gaps and move information between the blocks. The complete Simulink model is presented in figure 45.

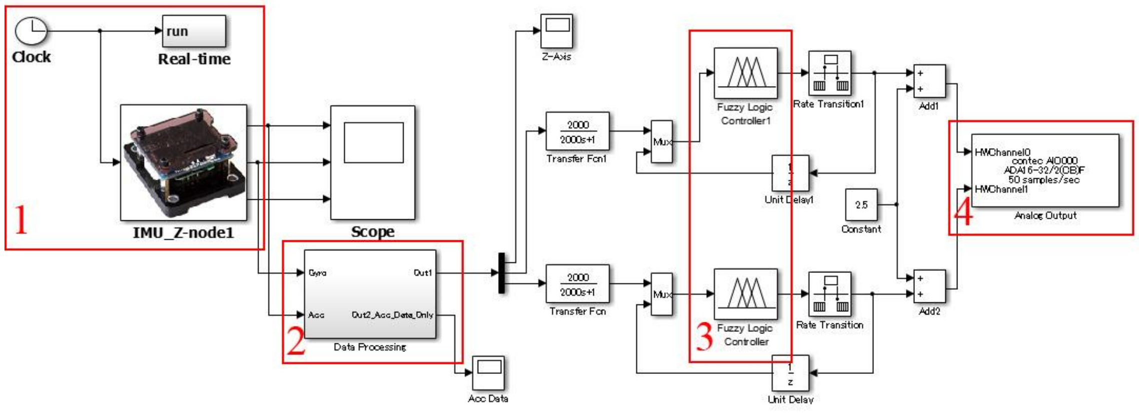


Fig. 45. The complete control system for EPW

A better view of the Simulink model can be found in appendix 14.

In the figure 45 there are four areas marked. They are the areas described in past four sections. Area 1 is the ZMP IMU-Z MATLAB model. Area 2 is the block that contains the processing of measurement data. Area 3 includes two fuzzy logic controllers. The top one controls the turning of the EPW. The bottom controller handles the forward and backward movement. Area 4 is the analog output block that is in charge of transfering the control signals to the EPW through the I/O card.

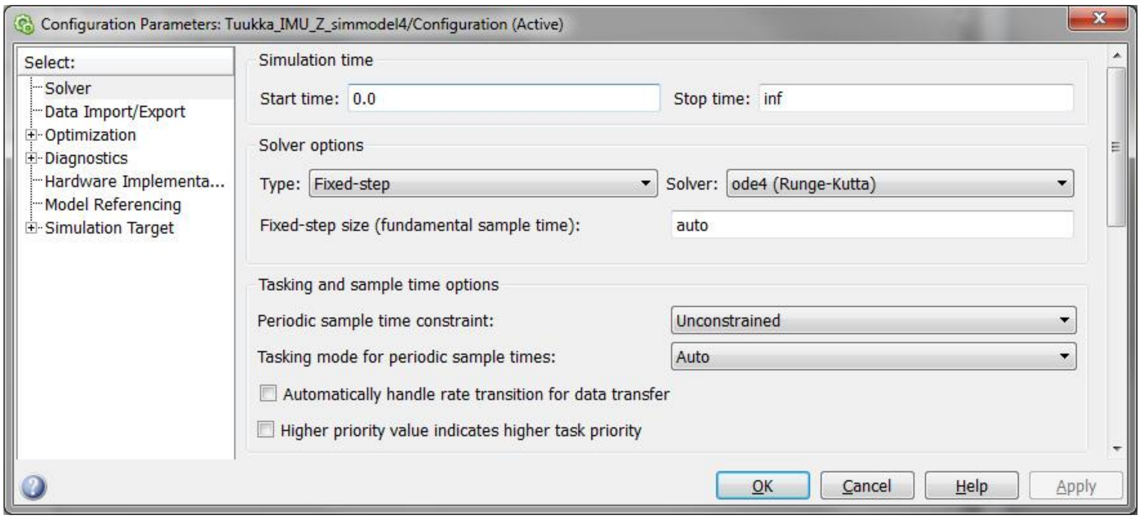
As mentioned before the areas 1 and 2 were pre-made tools that were accessible when this project was carried out. Between these areas there are simply wires connected that transfer gyro and acceleration data from IMU-Z into the data processing block (area 2).

The signal that comes from output port of data processing block contains 3-axis information of position of the IMU-Z. Three dimensional information is not needed this time. Two dimensional data is suitable for the purpose of controlling the EPW. Separating the needed signals can be done using a demux block. These signals that contain only 1-axis information need to be transfered into area 3.

Area 3 contains the fuzzy logic controllers of the control system. Feedback connections were added to both fuzzy controllers. The feedback signals go through mux blocks to join the feedback signal together with signals coming from area 2. Even after the mux blocks the signals can be processed separately inside the fuzzy logic controller blocks. Feedback signals have one step delay.

In section 3.1 there was mentioned that each axis of the joystick control work in range of 0...+5 V. Signals coming out of fuzzy controllers are in range of -2.5...+2.5 V. Because of this a constant is needed. A constant of 2.5 is added to each of the two signals coming the fuzzy controllers. This way the signals are in needed range. Now the signals are acceptable to be transferred to the EPW through the I/O card.

The used simulation parameters can be seen in figure 46.


Configuration Parameters: Tuukka_IMU_Z_simmodel4/Configuration (Active)

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
- Hardware Implementa...
- Model Referencing
- Simulation Target

Simulation time

Start time: 0.0 Stop time: inf

Solver options

Type: Fixed-step Solver: ode4 (Runge-Kutta)

Fixed-step size (fundamental sample time): auto

Tasking and sample time options

Periodic sample time constraint: Unconstrained

Tasking mode for periodic sample times: Auto

☐ Automatically handle rate transition for data transfer

☐ Higher priority value indicates higher task priority

Fig. 46. Simulation parameters of the Simulink program

5 SUMMARY

First big step in this project was choosing the platform to develop the control system on. MATLAB and Simulink was chosen instead of using a mere programming language. Simulink is a graphical editor and therefore gives a complete overview of the whole control system in just a few windows. Simulink served the purpose of building and tuning the system very well in this project.

Preparing the equipment consisted of assembling the I/O system, connecting the cables and installing drivers for used devices. Some measurements were made in order to design the control system to feed right kind of signals into the wheelchair through the I/O card.

The control system was built in Simulink environment. Commercially available libraries were used for communication between the motion sensor and MATLAB. Fuzzy logic controllers were built from scratch. Fuzzy control system suited this system well and it was easy to build.

Using the EPW is very simple. A single M-file starts the user interface and Simulink model. Communication settings can be selected from the user interface. Also the Simulink program can be started from the user interface. As the program is running and the IMU-Z is switched on the EPW can be controlled using the hands-free system.

Because the control system is operated using the angular changes in the IMU-Z, the system is relatively flexible of the placement of the IMU-Z. It could be attached to the head of a EPW user when the EPW movements correspond to the head movements. For example turning a head turns the EPW. A little nod of a head could make the EPW move forward.

When testing the built control system the performance of the system left a lot to be desired. It worked as required but was very slow. Tuning the Simulink program did not seem to have much effect on the performance. Due to time restrictions of the project testing and tuning were left to very minimum. The built control system along with created files were passed on to next person who would take responsibility of further development of this project.

6 REFERENCES

1. Golden Technologies 2011. Golden Technologies Mobility Products | Alante DX. Available at: http://www.goldentech.com/products/alante-dx/. Accessed May 1st, 2013.

2. Wikipedia contributors 2011. Motorized wheelchair. Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/w/index.php?title=Motorized_wheelchair&oldid=469638234. Accessed January 17th, 2012.

3. Wikipedia contributors 2011. Motion capture. Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/w/index.php?title=Motion_capture&oldid=469188812. Accessed January 18th, 2012.

4. Shi, Huxia 2009. Motion Tracking in VR. York University. Available at: http://www.cse.yorku.ca/~huxiashi/6335.ppt. Accessed December 13th, 2011.

5. Wikipedia contributors 2013. Inertial measurement unit. Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/w/index.php?title=Inertial_measurement_unit&oldid=550602245 . Accessed May 1st, 2013.

6. Zentrum für Sensorsysteme 2012. Navigation. Available at: http://www.zess.uni-siegen.de/home/das-zess/forschung/navigation.html. Accessed: March 12th, 2012.

7. Lammila, Mika & Karhu, Otso 2007. CAN ja CANopen -perusteet.

8. Alanen, Jarmo 2000. CAN-ajoneuvojen ja koneiden sisäinen paikallisväylä. VTT Automaatio, Koneautomaatio.

9. Wikipedia contributors 2013. CAN bus. Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/w/index.php?title=CAN_bus&oldid=552095285. Accessed April 28th, 2013.

10. Wikipedia contributors 2013. Fuzzy control system. Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/w/index.php?title=Fuzzy_control_system&oldid=540419607. Accessed March 21st, 2013.

11. Goebel, Greg 2003. Introduction to fuzzy logic & fuzzy control. Available at: http://www.faqs.org/docs/fuzzy/. Accessed March 21st, 2013.

12. Yamaha 2004. ヤハマ・タウニィジョイ軽量型電動車イス取り扱説明書. Yahama Towny Joy User’s manual.

13. ZMP INC. 2011. ZMP IMU-Z Manual: Overview.

14. Lawicel AB 2013. CANUSB. Available at: http://www.can232.com/?page_id=16. Accessed April 28th, 2013.

15. CONTEC Co., Ltd. 2011. High Resolution & Speed Analog I/O Card ADA16-32/2(CB)F User's Guide.

16. CONTEC Co., Ltd. 2006. Screw Terminal DTP-64(PC) User's Guide.

17. CONTEC Co., Ltd. 2009. Data Acquisition Library for MATLAB - ML-DAQ Setup & Reference Guide.

18. ZMP INC. 2013. 9 軸（加速度・ジャイロ・地磁気）ワイヤレスモーションセンサ＆SDK e-nuvo IMU-Z2 | MATLAB/Simulink 対応. Available at: http://www.zmp.co.jp/imu-z_matlab.html. Accessed May 1st, 2013.

Base data frame format.

Field	Length	Purpose
Start-of-Frame	1 bit	Starter bit of a CAN-message frame
ID	11 bits	Identifier used to determine the priority.
Remote transmission request	1 bit	Defines whether a message is a normal data message or a transmission request.
IDE	1 bit	Identifier extension bit.
Reserved bit (r0)	1 bit	
Data length code	4 bits	Length of the data in the message.
Data	0-8 bytes	The actual data sent to a device.
Cyclic redundancy check (CRC)	15 bits	Checksum of the message. If the checksum fails the message is rejected.
CRC delimiter	1 bit	
ACK slot	1 bit	Acknowledgement of a received message.
ACK delimiter	1 bit	
End-of-Frame	7 bits	Bits that tell a device that the message has come to an end.

Extended data frame format.

Field	Length	Purpose
Start-of-Frame	1 bit	Starter bit of a CAN-message frame
ID A	11 bits	First part of the identifier used to determine the priority.
Substitute remote request	1 bit	
IDE	1 bit	Identifier extension bit.
ID B	18 bits	Second part of the identifier used to determine the priority.
Remote transmission request	1 bit	Defines whether a message is a normal data message or a transmission request.
Reserved bits (r0 & r1)	2 bits	
Data length code	4 bits	Length of the data in the message.
Data	0-8 bytes	The actual data sent to a device.
Cyclic redundancy check (CRC)	15 bits	Checksum of the message. If the checksum fails the message is rejected.
CRC delimiter	1 bit	
ACK slot	1 bit	Acknowledgement of a received message.
ACK delimiter	1 bit	
End-of-Frame	7 bits	Bits that tell a device that the message has come to an end.

Specifications of Yamaha Towny Joy.

Specifications		
Product name		Towny Joy
Drive system		Rear-wheel drive
Dimensions (length x width x height)		1000 x 550 x 900 mm
Weight		22 kg (battery included)
Steering method	Self	Joystick
	Caregiver	Manual steering
Driving wheels		16"
Caster wheels		7"
Control system		Microcomputer
Braking system		Electromagnetic brakes, engine braking
Motors		24 V 90 W
Top speed	Drive	Low gear 2.5 km/h, High gear 4.5 km/h
	Reverse	2.0 km/h
Steepest climbable slope		6°
Battery		NiMH, 2.9 kg
		24 V 6.7 Ah
Battery charger	Input	AC 100 V ~ 240 V 50/60 Hz
	Charging system	Fully automatic
	Charging time	2~3 hours (room temperature)
Range with one recharge		10 km
Frame	Material	Reinforced aluminum
	Armrest	Removable
	Footrest	Removable

Specifications of ZMP IMU-Z motion sensor.

Specifications		
Product name		ZMP IMU-Z
Acceleration sensor		3-axis ± 2 g, ± 4 g, ± 8 g
Gyroscopic sensor		3-axis ± 500 °/s, ± 2000 °/s
Compass sensor		3-axis ± 0.7 G ~ ± 4.5 G
Connections		Bluetooth, CAN
Sampling rate		10 ms ~ 10 s, 10 ms steps
Dimensions	Length	42 mm
	Width	52.5 mm
	Height	20.5 mm
Weight		35 g
Maximum number of sensors at a time		28

```
IMU-Z CAN message format
ver 1.3
2010-06-30
SegaWa.


========================================
CAN_ID
----------------------------------------
             CAN-ID (11)
             +------------+-------------+---------------+
             | NODE_NO(5) | DIRECTION(1) | MESSAGE_ID(5) |
             +------------+-------------+---------------+

             NODE_NO
              0 : host
              1-28 : IMU-Z
              29 : reserve
              30 : reserve
              31 : all

             DIRECTION
              0 : Slave <- Host, Slave <- Master (Command)
              1 : Slave -> Master, Slave -> Host (Respose, Data)



========================================
MESSAGE_ID
----------------------------------------

             ECHO (target_no)(0x0)(0x01)
              +-----------+
              +-----------+

             ECHO (sender_no)(0x1)(0x01)
              +-----------+
              +-----------+

             MEASUREMENT_ACC (sender_no)(0x1)(0x02)
              +-----------+
             0| time_h
             1| time_l
             2| acc_x_h
             3| acc_x_l
             4| acc_y_h
             5| acc_y_l
             6| acc_z_h
             7| acc_z_l
              +-----------+

             MEASUREMENT_GYRO (sender_no)(0x1)(0x03)
              +-----------+
             0| time_h
             1| time_l
             2| gyro_x_h
             3| gyro_x_l
             4| gyro_y_h
             5| gyro_y_l
             6| gyro_z_h
             7| gyro_z_l
              +-----------+
             MEASUREMENT_COMPASS (sender_no)(0x1)(0x04)
              +-----------+
             0| time_h
             1| time_l
             2| comp_x_h
             3| comp_x_l
             4| comp_y_h
             5| comp_y_l
             6| comp_z_h
             7| comp_z_l
              +-----------+

             STATUS (target_no)(0x0)(0x05)
              +-----------+
              +-----------+

             STATUS (sender_no)(0x1)(0x05)
              +-----------+
             0| role
             1| period_h
             2| period_l
             3| range_acc(2) range_gyro(2) range_comp(3)
             4| batt
             5| msg_foramt
              +-----------+

             SET_ROLE (target_no)(0x0)(0x06)
              +-----------+
             0| role
              +-----------+

             SET_PERIOD (target_no)(0x0)(0x07)
              +-----------+
             0| period_h
             1| period_l
              +-----------+

             SET_RANGE (target_no)(0x0)(0x08)
              +-----------+
             0| range_acc(2) range_gyro(2) range_comp(3)
              +-----------+

             SET_NODE_NO (target_no)(0x0)(0x09)
              +-----------+
             0| node_no
              +-----------+

             SAVE (target_no)(0x0)(0x0a)
              +-----------+
              +-----------+

             RESET_TIMESTAMP (target_no)(0x0)(0x0b)
              +-----------+
              +-----------+

             SET_MEASUREMENT_STATE (target_no)(0x0)(0x0c)
              +-----------+
             0| enable(1)
              +-----------+

             DEVICEP_PROFILE (target_no)(0x0)(0x0d)
              +-----------+
              +-----------+
                          DEVICEP_PROFILE (sender_no)(0x1)(0x0d)
                           +-----------+
                          0| hardware
                          1| firmware
                          2| btaddr[0]
                          3| btaddr[1]
                          4| btaddr[2]
                          5| btaddr[3]
                          6| btaddr[4]
                          7| btaddr[5]
                           +-----------+

                          FACTORY_RESET (target_no)(0x0)(0x0e)
                           +-----------+
                           +-----------+

                          SET_BINARY (target_no)(0x0)(0x0f)
                           +-----------+
                          0| enable(1)
                           +-----------+

             ========================================
             ID & MASK
             ----------------------------------------

                          Single node
                                       *config mode
                                                                mask   11111|1|00000
                                                    slot0 Receive req command
                                                                     (my_no)|0|XXXXX
                                                    slot1 Receive broadcast req command
                                                                     (bc_no)|0|XXXXX
                                                    slot8 Send response of command
                                                                     (my_no)|1|XXXXX
                                       *run mode
                                                                mask   11111|1|00000

                          Master node
                                       *config mode
                                                                mask   11111|1|00000
                                                    slot0 Receive req command
                                                                     (my_no)|0|XXXXX
                                                    slot1 Receive broadcast req command
                                                                     (bc_no)|0|XXXXX
                                                    slot8 Send response of command
                                                                     (my_no)|1|XXXXX

                                       *run mode
                                                                mask   XXXXX|1|XXXXX
                                                    slot2 Collect all report
                                                                       XXXXX|1|XXXXX
```

```
Slave node
             *config mode
                                        mask   11111|1|00000
                          slot0 Receive req command
                                             (my_no)|0|XXXXX
                          slot1 Receive broadcast req command
                                             (bc_no)|0|XXXXX
                          slot8 Send response of command
                                             (my_no)|1|XXXXX
             *run mode
                                        mask   11111|1|00000
                          slot0 Receive req command
                                             (my_no)|0|XXXXX
                          slot1 Receive broadcast req command
                                             (bc_no)|0|XXXXX
                          slot8 Send response of command
                                             (my_no)|1|XXXXX
                          slot9 Send report
                                             (my_no)|1|XXXXX
```

Lawicel AB CANUSB

Dimensions:

- Length: 55 mm
- Width: 36 mm
- Height: 16 mm

Specifications:

- CAN bitrate of up to 1Mbit/s
- Functional in industril temperature range -40°C…+85°C
- USB 2.0 full speed using FTDI FT245RL chip
- Philips SJA1000 CAN controller running at 16 MHz
- ISO 11898-24V compatible Philips 82C251 CAN transceiver
- Complies with CAN 2.0A (11-bit IDs) and CAN 2.0B (29-bit IDs)
- RTR frame support
- 32 CAN frames deep FIFO buffer
- CiA DS102-1 standard compatible
- LED lights to indicate activity and errors
- Supports Windows and Linux platforms

```
% IMUZ_START.m

close all
clear all
clc

global portName Imuz_node_no Imuz_time Imuz_Acc1 Imuz_Acc2 Imuz_Acc3
Imuz_gyro1 Imuz_gyro2 Imuz_gyro3 Imuz_comp1 Imuz_comp2 Imuz_comp3
global plot1 plot2 plot3 ui_8 ui_4 ui_6  CB_old bode_old ui_10 ui_12
ui_17
global line1_1 line1_2 line1_3 line2_1 line2_2 line2_3 line3_1 line3_2
line3_3
global ui_Acc1 ui_Acc2 ui_Acc3 ui_Gyro1 ui_Gyro2 ui_Gyro3 ui_Comp1
ui_Comp2 ui_Comp3
global OUT_data nodeNo_plot save_data dll_path


simmodel='IMU_Z_simmodel';
open([simmodel,'.mdl']);
stime=0.1;%%simulation time
% clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dll_path='C:\Program Files\ZMP\IMU-Z MATLAB Connection';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dll_path=[dll_path,'\'];


portName='';
Imuz_node_no =0;
Imuz_time=0;
Imuz_Acc1=0; Imuz_Acc2=0; Imuz_Acc3=0;
Imuz_gyro1=0; Imuz_gyro2=0; Imuz_gyro3=0;
Imuz_comp1=0; Imuz_comp2=0; Imuz_comp3=0;

for i=1:28
    OUT_data(i).nodeNo=[];
    OUT_data(i).time=[];
    OUT_data(i).data=[];
    OUT_data(i).time_off=[];
    OUT_data(i).save_start_time=[];
    OUT_data(i).save_stop_time=[];
    OUT_data(i).save_start_ren=[];
    OUT_data(i).save_stop_ren=[];
end
nodeNo_plot=1;

save_data.start_swith=0;
save_data.stop_swith=0;
save_data.node_length=[];
save_data.time_real=0;
save_data.save_path=cd;
save_data.label_box={'NodeNo','time','AccX','AccY','AccZ','GyroX','Gyr
oY','GyroZ','CompX','CompY','CompZ'};
save_Data.save_node_old={};

%%
fig_handle=figure(100);
set(fig_handle,'Position',[30 200 690 570]);
set(fig_handle,'Color',[0.92549 0.913725 0.847059]);
set(fig_handle,'Menubar','none')
set(fig_handle,'name','IMU_Z Viewer GUI');
```

```
%%
ui_id_frame=uicontrol('style','frame');
set(ui_id_frame,'position',[15 525 290 40]);
set(ui_id_frame,'BackgroundColor',[0.92549 0.913725 0.847059]);

ui_comp_frame=uicontrol('style','frame');
set(ui_comp_frame,'position',[5 217 680 2]);
set(ui_comp_frame,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(ui_comp_frame,'ForegroundColor',[0.75 0.75 0.75]);

ui_gyro_frame=uicontrol('style','frame');
set(ui_gyro_frame,'position',[5 367 680 2]);
set(ui_gyro_frame,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(ui_gyro_frame,'ForegroundColor',[0.75 0.75 0.75]);

ui_Acc_frame=uicontrol('style','frame');
set(ui_Acc_frame,'position',[5 515 680 2]);
set(ui_Acc_frame,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(ui_Acc_frame,'ForegroundColor',[0.75 0.75 0.75]);

ui_save_frame=uicontrol('style','frame');
set(ui_save_frame,'position',[5 70 680 2]);
set(ui_save_frame,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(ui_save_frame,'ForegroundColor',[0.75 0.75 0.75]);

ui_save_frame=uicontrol('style','frame');
set(ui_save_frame,'position',[15 10 660 45]);
set(ui_save_frame,'BackgroundColor',[0.92549 0.913725 0.847059]);

%%
ui_1=uicontrol('style','pushbutton');
set(ui_1,'position',[320 530 80 30]);
set(ui_1,'string','START');
set(ui_1,'Callback','Imuz_CAN_start;');

ui_2=uicontrol('style','pushbutton');
set(ui_2,'position',[400 530 80 30]);
set(ui_2,'string','STOP');
set(ui_2,'Callback','Imuz_CAN_stop');

ui_3=uicontrol('style','text');
set(ui_3,'string','Mode:');
set(ui_3,'position',[20 530 70 20]);
set(ui_3,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(ui_3,'HorizontalAlignment','left');

CB_old={};
CB_old(1)={'CAN'};
CB_old(2)={'Bluetooth'};

ui_4=uicontrol('style','popu');
set(ui_4,'position',[80 485 80 70]);
set(ui_4,'string',CB_old);
set(ui_4,'BackgroundColor',[1 1 1]);

ui_5=uicontrol('style','text');
set(ui_5,'string','Board:');
set(ui_5,'position',[170 530 70 20]);
set(ui_5,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(ui_5,'HorizontalAlignment','left');
```

```
bode_old={};
for i=1:40
    eval(['bode_old(',num2str(i),')={''COM',num2str(i),'''};']);
end

ui_6=uicontrol('style','popu');
set(ui_6,'position',[220 485 80 70]);
set(ui_6,'string',bode_old);
set(ui_6,'BackgroundColor',[1 1 1]);

ui_7=uicontrol('style','text');
set(ui_7,'string','Node No:');
set(ui_7,'position',[495 530 70 20]);
set(ui_7,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(ui_7,'HorizontalAlignment','left');

node_old={};
for i=1:28
    eval(['node_old(',num2str(i),')={''',num2str(i),'''};']);
end

ui_8=uicontrol('style','popu');
set(ui_8,'position',[550 485 120 70]);
set(ui_8,'string',node_old);
set(ui_8,'BackgroundColor',[1 1 1]);

%% data save

ui_9=uicontrol('style','text');
set(ui_9,'position',[30 40 60 20]);
set(ui_9,'string','data save')
set(ui_9,'FontWeight',' bold');

ui_10=uicontrol('style','edit');
set(ui_10,'position',[100 20 40 20]);
% set(ui_10,'Enable','Off');
set(ui_10,'Enable','On');
set(ui_10,'string',[]);

ui_11=uicontrol('style','text');
set(ui_11,'position',[145 15 10 20]);
set(ui_11,'string','?`')
set(ui_11,'Enable','Off');

ui_12=uicontrol('style','edit');
set(ui_12,'position',[160 20 40 20]);
% set(ui_12,'Enable','Off');
set(ui_12,'Enable','On');
set(ui_12,'string',[]);

ui_13=uicontrol('style','pushbutton');
set(ui_13,'position',[210 15 50 30]);
set(ui_13,'string','START');
set(ui_13,'Callback','for
i=save_data.node_length;OUT_data(i).save_start_time=[];OUT_data(i).sav
e_stop_time=[];end;save_data.start_swith=1;save_data.stop_swith=0;');

ui_14=uicontrol('style','pushbutton');
set(ui_14,'position',[260 15 50 30]);
set(ui_14,'string','STOP');
set(ui_14,'Callback','save_data.start_swith=0;save_data.stop_swith=1;'
);

ui_15=uicontrol('style','pushbutton');
set(ui_15,'position',[310 15 50 30]);
set(ui_15,'string','RESET');
set(ui_15,'Callback','save_data.start_swith=0;save_data.stop_swith=0;'
);

ui_16=uicontrol('style','pushbutton');
set(ui_16,'position',[580 15 90 30]);
set(ui_16,'string','save Data');
set(ui_16,'Callback','Imuz_data_save');

save_node_old={};
save_node_old(1)={'All'};
for i=1:28
    eval(['save_node_old(',num2str(i+1),')={''',num2str(i),'''};']);
end
save_data.save_node_old=save_node_old;

ui_17=uicontrol('style','popu');
set(ui_17,'position',[470 10 110 30]);
set(ui_17,'string',save_data.save_node_old);
set(ui_17,'BackgroundColor',[1 1 1]);

ui_18=uicontrol('style','text');
set(ui_18,'string','save Node No:');
set(ui_18,'position',[390 15 80 20]);
set(ui_18,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(ui_18,'HorizontalAlignment','left');

ui_19=uicontrol('style','text');
set(ui_19,'string','save Time [s]:');
set(ui_19,'position',[20 15 80 20]);
set(ui_19,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(ui_19,'HorizontalAlignment','left');



%%
% %%%%%%%%%%%%%%%Imuz_Acc1 Imuz_Acc2 Imuz_Acc3
ui_Acc1_txt=uicontrol('style','text');
set(ui_Acc1_txt,'position',[500 460 50 20]);
set(ui_Acc1_txt,'string','AccX')
set(ui_Acc1_txt,'HorizontalAlignment','left');

ui_Acc2_txt=uicontrol('style','text');
set(ui_Acc2_txt,'position',[500 430 50 20]);
set(ui_Acc2_txt,'string','AccY')
set(ui_Acc2_txt,'HorizontalAlignment','left');

ui_Acc3_txt=uicontrol('style','text');
set(ui_Acc3_txt,'position',[500 400 50 20]);
set(ui_Acc3_txt,'string','AccZ')
set(ui_Acc3_txt,'HorizontalAlignment','left');
%%unit[G]
ui_Acc1_txt=uicontrol('style','text');
set(ui_Acc1_txt,'position',[630 460 50 20]);
set(ui_Acc1_txt,'string','[g]')
set(ui_Acc1_txt,'HorizontalAlignment','left');
```

```
ui_Acc2_txt=uicontrol('style','text');
set(ui_Acc2_txt,'position',[630 430 50 20]);
set(ui_Acc2_txt,'string','[g]')
set(ui_Acc2_txt,'HorizontalAlignment','left');

ui_Acc3_txt=uicontrol('style','text');
set(ui_Acc3_txt,'position',[630 400 50 20]);
set(ui_Acc3_txt,'string','[g]')
set(ui_Acc3_txt,'HorizontalAlignment','left');

ui_Acc1=uicontrol('style','edit');
set(ui_Acc1,'position',[540 465 90 18]);
set(ui_Acc1,'string',Imuz_Acc1)
set(ui_Acc1,'BackgroundColor',[1,1,1]);

ui_Acc2=uicontrol('style','edit');
set(ui_Acc2,'position',[540 435 90 18]);
set(ui_Acc2,'string',Imuz_Acc2)
set(ui_Acc2,'BackgroundColor',[1,1,1]);

ui_Acc3=uicontrol('style','edit');
set(ui_Acc3,'position',[540 405 90 18]);
set(ui_Acc3,'string',Imuz_Acc3)
set(ui_Acc3,'BackgroundColor',[1,1,1]);

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%Imuz_gyro1 Imuz_gyro2 Imuz_gyro3
ui_Gyro1_txt=uicontrol('style','text');
set(ui_Gyro1_txt,'position',[500 315 50 20]);
set(ui_Gyro1_txt,'string','GyroX')
set(ui_Gyro1_txt,'HorizontalAlignment','left');

ui_Gyro2_txt=uicontrol('style','text');
set(ui_Gyro2_txt,'position',[500 285 50 20]);
set(ui_Gyro2_txt,'string','GyroY')
set(ui_Gyro2_txt,'HorizontalAlignment','left');

ui_Gyro3_txt=uicontrol('style','text');
set(ui_Gyro3_txt,'position',[500 255 50 20]);
set(ui_Gyro3_txt,'string','GyroZ')
set(ui_Gyro3_txt,'HorizontalAlignment','left');

ui_Gyro1_txt=uicontrol('style','text');
set(ui_Gyro1_txt,'position',[630 315 70 20]);
set(ui_Gyro1_txt,'string','[deg/s]')
set(ui_Gyro1_txt,'HorizontalAlignment','left');

ui_Gyro2_txt=uicontrol('style','text');
set(ui_Gyro2_txt,'position',[630 285 70 20]);
set(ui_Gyro2_txt,'string','[deg/s]')
set(ui_Gyro2_txt,'HorizontalAlignment','left');

ui_Gyro3_txt=uicontrol('style','text');
set(ui_Gyro3_txt,'position',[630 255 70 20]);
set(ui_Gyro3_txt,'string','[deg/s]')
set(ui_Gyro3_txt,'HorizontalAlignment','left');

ui_Gyro1=uicontrol('style','edit');
set(ui_Gyro1,'position',[540 320 90 18]);
set(ui_Gyro1,'string',Imuz_gyro1)
set(ui_Gyro1,'BackgroundColor',[1,1,1]);

ui_Gyro2=uicontrol('style','edit');
set(ui_Gyro2,'position',[540 290 90 18]);
set(ui_Gyro2,'string',Imuz_gyro2)
set(ui_Gyro2,'BackgroundColor',[1,1,1]);

ui_Gyro3=uicontrol('style','edit');
set(ui_Gyro3,'position',[540 260 90 18]);
set(ui_Gyro3,'string',Imuz_gyro3)
set(ui_Gyro3,'BackgroundColor',[1,1,1]);

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%Imuz_comp1 Imuz_comp2 Imuz_comp3
ui_Comp1_txt=uicontrol('style','text');
set(ui_Comp1_txt,'position',[500 165 50 20]);
set(ui_Comp1_txt,'string','CompX')
set(ui_Comp1_txt,'HorizontalAlignment','left');

ui_Comp2_txt=uicontrol('style','text');
set(ui_Comp2_txt,'position',[500 135 50 20]);
set(ui_Comp2_txt,'string','CompY')
set(ui_Comp2_txt,'HorizontalAlignment','left');

ui_Comp3_txt=uicontrol('style','text');
set(ui_Comp3_txt,'position',[500 105 50 20]);
set(ui_Comp3_txt,'string','CompZ')
set(ui_Comp3_txt,'HorizontalAlignment','left');

ui_Comp1_txt=uicontrol('style','text');
set(ui_Comp1_txt,'position',[630 165 50 20]);
set(ui_Comp1_txt,'string','[G]')
set(ui_Comp1_txt,'HorizontalAlignment','left');

ui_Comp2_txt=uicontrol('style','text');
set(ui_Comp2_txt,'position',[630 135 50 20]);
set(ui_Comp2_txt,'string','[G]')
set(ui_Comp2_txt,'HorizontalAlignment','left');

ui_Comp3_txt=uicontrol('style','text');
set(ui_Comp3_txt,'position',[630 105 50 20]);
set(ui_Comp3_txt,'string','[G]')
set(ui_Comp3_txt,'HorizontalAlignment','left');

ui_Comp1=uicontrol('style','edit');
set(ui_Comp1,'position',[540 170 90 18]);
set(ui_Comp1,'string',Imuz_comp1)
set(ui_Comp1,'BackgroundColor',[1,1,1]);

ui_Comp2=uicontrol('style','edit');
set(ui_Comp2,'position',[540 140 90 18]);
set(ui_Comp2,'string',Imuz_comp2)
set(ui_Comp2,'BackgroundColor',[1,1,1]);

ui_Comp3=uicontrol('style','edit');
set(ui_Comp3,'position',[540 110 90 18]);
set(ui_Comp3,'string',Imuz_comp3)
set(ui_Comp3,'BackgroundColor',[1,1,1]);
```

```
%%%%%%%%%%%%%%%%
plot1=axes('position',[.10  .71  .60  .15]);
    line1_1=plot(0,0,'r+-','LineWidth',2);hold on
    line1_2=plot(0,0,'g+-','LineWidth',2);hold on
    line1_3=plot(0,0,'b+-','LineWidth',2);hold off
    line1_legend=legend([line1_1,line1_2,line1_3],'X','Y','Z',1);
    set(line1_legend,'Orientation','horizontal');
    set(line1_legend,'position',[0.1007    0.8240    0.2671
0.0351])
    xlabel('Time [s]');ylabel('Acceleration [g]');grid on
    set(plot1,'ylim',[-2 2]);
    set(plot1,'xlim',[0 7]);

plot2=axes('position',[.10  .45  .60  .15]);
    line2_1=plot(0,0,'r+-','LineWidth',2);hold on
    line2_2=plot(0,0,'g+-','LineWidth',2);hold on
    line2_3=plot(0,0,'b+-','LineWidth',2);hold off
    line2_legend=legend([line2_1,line2_2,line2_3],'X','Y','Z');
    set(line2_legend,'Orientation','horizontal');
    set(line2_legend,'position',[0.1007    0.5646    0.2671
0.0351])
    xlabel('Time [s]');ylabel('Angular velocity[deg/s]');grid on
    set(plot2,'ylim',[-1000 1000]);
    set(plot2,'xlim',[0 7]);

plot3=axes('position',[.10  .19  .60  .15]);
    line3_1=plot(0,0,'r+-','LineWidth',2);hold on
    line3_2=plot(0,0,'g+-','LineWidth',2);hold on
    line3_3=plot(0,0,'b+-','LineWidth',2);hold off
    line3_legend=legend([line3_1,line3_2,line3_3],'X','Y','Z');
    set(line3_legend,'Orientation','horizontal');
    set(line3_legend,'position',[0.1007    0.3049    0.2671
0.0351])
    xlabel('Time [s]');ylabel('Magnetic field [G]');grid on
    set(plot3,'ylim',[-0.7 0.7]);
    set(plot3,'xlim',[0 7]);

%%plot application screen
plot1_txt=uicontrol('style','text');
set(plot1_txt,'string','Accelerometer');
set(plot1_txt,'position',[260 495 90 15]);
set(plot1_txt,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(plot1_txt,'HorizontalAlignment','left');
set(plot1_txt,'FontWeight',' bold');

plot2_txt=uicontrol('style','text');
set(plot2_txt,'string','Gyroscope');
set(plot2_txt,'position',[260 345 90 15]);
set(plot2_txt,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(plot2_txt,'HorizontalAlignment','left');
set(plot2_txt,'FontWeight',' bold');

plot3_txt=uicontrol('style','text');
set(plot3_txt,'string','Compass');
set(plot3_txt,'position',[260 195 90 15]);
set(plot3_txt,'BackgroundColor',[0.92549 0.913725 0.847059]);
set(plot3_txt,'HorizontalAlignment','left');
set(plot3_txt,'FontWeight',' bold');

%%%%%%%%%%%%%%%
```

```
% waitfortreal.m



function waitfortreal(tsim)
% This function is intended to synchronize the Simulink clock with
real time.
% When called with t = 0, the beginning time is established.
Subsequent calls, with
% tsim > 0, enter a busy wait state until the real-time clock catches
up to tsim.
% It is assumed that the Simulink clock, tsim, is faster than real
time.

%
%   Stan Quinn December 1996
%   Copyright (c) 1995-97 by The MathWorks, Inc.
% $Revision: 1.1 $
%

global waitfortrealTstart
time = clock;
if (tsim == 0)
   waitfortrealTstart = time;
else
   while etime(clock, waitfortrealTstart) < tsim
   end
end
```

```
% sfun_Imuz_sp.m

function sfun_Imuz_sp(block)
setup(block);

function setup(block)
  block.NumDialogPrms     = 0;

  block.NumInputPorts  = 1;
  block.NumOutputPorts = 3;

  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;

    % Override input port properties
  block.InputPort(1).DatatypeID  = 0;  % double
  block.InputPort(1).Complexity  = 'Real';
  block.InputPort(1).DirectFeedThrough = true;
  block.InputPort(1).SamplingMode = 'Sample';
  block.InputPort(1).Dimensions   = 1;

  block.OutputPort(1).Complexity   = 'Real';
  block.OutputPort(1).DataTypeId   = 0;
  block.OutputPort(1).SamplingMode = 'Sample';
  block.OutputPort(1).Dimensions   = 3;

  block.OutputPort(2).Complexity   = 'Real';
  block.OutputPort(2).DataTypeId   = 0;
  block.OutputPort(2).SamplingMode = 'Sample';
  block.OutputPort(2).Dimensions   = 3;

  block.OutputPort(3).Complexity   = 'Real';
  block.OutputPort(3).DataTypeId   = 0;
  block.OutputPort(3).SamplingMode = 'Sample';
  block.OutputPort(3).Dimensions   = 3;

  block.SampleTimes = [0 0];

  %% ------------------------------------------------------------------
  % M-file
  block.SetAccelRunOnTLC(false);
  block.RegBlockMethod('Start', @Start);
  %%
  block.RegBlockMethod('Outputs', @Outputs);
  %%
  block.RegBlockMethod('Terminate', @Terminate);

function Start(block)
    global portName ui_4 ui_6 CB_old bode_old dll_path
    kk1=get(ui_4,'Value');
    kk2=get(ui_6,'Value');
    portName=[char(CB_old(kk1)),'(', char(bode_old(kk2)),')'];

    if libisloaded([dll_path,'ImuzCommunication'])
    else

loadlibrary([dll_path,'ImuzCommunication.dll'],[dll_path,'mat_ImuzComm
unication.h'])
    end
    if libisloaded([dll_path,'SimpleCom'])
```

```
   else

loadlibrary([dll_path,'SimpleCom.dll'],[dll_path,'mat_SimpleCom.h'])
    end
    if libisloaded([dll_path,'ImuzMatInterface'])
    else
        loadlibrary([dll_path,'ImuzMatInterface.dll'],
[dll_path,'mat_ImuzMatInterface.h'])
    end
    calllib('ImuzMatInterface', 'ImuzIf_Open', portName);

%endfunction

function Outputs(block)
global Imuz_Acc1 Imuz_Acc2 Imuz_Acc3 Imuz_gyro1 Imuz_gyro2 Imuz_gyro3
Imuz_comp1 Imuz_comp2 Imuz_comp3
global nodeNo_plot ui_10 ui_12 ui_17
global plot1 plot2 plot3
global line1_1 line1_2 line1_3 line2_1 line2_2 line2_3 line3_1 line3_2
line3_3
global Data reLen OUT_data ui_8 save_data
global ui_Acc1 ui_Acc2 ui_Acc3 ui_Gyro1 ui_Gyro2 ui_Gyro3 ui_Comp1
ui_Comp2 ui_Comp3

%%
%%%%--------------------GetRecentData---------------------
% %
    save_data.time_real = block.InputPort(1).Data;

    nodeNo_plot=get(ui_8,'Value');

    nodeNo=nodeNo_plot;

    st.node_no = 0;
    st.time = 0;
    st.acc(1)  = 0;  st.acc(2)  = 0;    st.acc(3)  = 0;
    st.gyro(1) = 0;  st.gyro(2) = 0;    st.gyro(3) = 0;
    st.comp(1) = 0;  st.comp(2) = 0;    st.comp(3) = 0;
%
    s = libstruct('ParamImuzMeasurement', st);
    calllib('ImuzMatInterface', 'ImuzIf_GetRecentData',nodeNo, s);

    Imuz_Acc_123=[double(s.acc(1)),double(s.acc(2)),double(s.acc(3))];

Imuz_gyro_123=[double(s.gyro(1)),double(s.gyro(2)),double(s.gyro(3))];

Imuz_comp_123=[double(s.comp(1)),double(s.comp(2)),double(s.comp(3))];

    block.OutputPort(1).Data = Imuz_Acc_123 ;
    block.OutputPort(2).Data = Imuz_gyro_123;
    block.OutputPort(3).Data = Imuz_comp_123;


%%
    %%%-----------------GetData-----------------------
    maxLen=10;
    reLen_1=0;
    pv_reLen = libpointer('int32Ptr', reLen_1);
    reLen = get(pv_reLen, 'Value');

    Data_1=[];
```

```
 for i=1:110
        Data_1=[Data_1,0];
    end
    pv = libpointer('singlePtr', Data_1);
    Data=get(pv, 'Value');

    calllib('ImuzMatInterface', 'ImuzIf_GetData',pv, pv_reLen,maxLen);
    Data=get(pv, 'Value');
    reLen=get(pv_reLen, 'Value');

    for i=1:reLen
        k=Data(11*(i-1)+1);
        if k==0
            break
        end
        OUT_data(k).data=[OUT_data(k).data;double(Data(11*(i-
1)+1:11*i))];
        OUT_data(k).nodeNo=k;
        OUT_data(k).time=OUT_data(k).data(1,2);
    end
    for i=1:28
        if size(OUT_data(i).nodeNo,1) ~= 0
            yebi_data=OUT_data(i).data;
            OUT_data(i).time_off=(yebi_data(:,2)-
OUT_data(i).time)./1000;
        else
            OUT_data(i).time_off=[];
        end
    end

    plot_data=double(OUT_data(nodeNo_plot).data);

    if size(plot_data,1) ~=0
        plot_data_time0=double(OUT_data(nodeNo_plot).time);
    else
        plot_data_time0=0;
        plot_data(1,1:11)=0;
    end

    Imuz_Acc1=plot_data(end,3); Imuz_Acc2=plot_data(end,4);
Imuz_Acc3=plot_data(end,5);
    Imuz_gyro1=plot_data(end,6); Imuz_gyro2=plot_data(end,7);
Imuz_gyro3=plot_data(end,8);
    Imuz_comp1=plot_data(end,9); Imuz_comp2=plot_data(end,10);
Imuz_comp3=plot_data(end,11);


set(ui_Acc1,'string',Imuz_Acc1);set(ui_Acc2,'string',Imuz_Acc2);set(ui
_Acc3,'string',Imuz_Acc3);

set(ui_Gyro1,'string',Imuz_gyro1);set(ui_Gyro2,'string',Imuz_gyro2);se
t(ui_Gyro3,'string',Imuz_gyro3);

set(ui_Comp1,'string',Imuz_comp1);set(ui_Comp2,'string',Imuz_comp2);se
t(ui_Comp3,'string',Imuz_comp3);

    %%%%%--------------------plot------------------
    time=(plot_data(:,2)-plot_data_time0)./1000;

    if time(end)>=7
        set(plot1,'xlim',[ time(end)-6,time(end)+1]);
        set(plot2,'xlim',[ time(end)-6,time(end)+1]);
    set(plot3,'xlim',[ time(end)-6,time(end)+1]);
    else
        set(plot1,'xlim',[0 7]);
        set(plot2,'xlim',[0 7]);
        set(plot3,'xlim',[0 7]);
    end

%     axes(plot1)
    set(line1_1,'XData',time,'YData',plot_data(:,3));
    set(line1_2,'XData',time,'YData',plot_data(:,4));
    set(line1_3,'XData',time,'YData',plot_data(:,5));

 %     axes(plot2)
    set(line2_1,'XData',time,'YData',plot_data(:,6));
    set(line2_2,'XData',time,'YData',plot_data(:,7));
    set(line2_3,'XData',time,'YData',plot_data(:,8));

%     axes(plot3)
    set(line3_1,'XData',time,'YData',plot_data(:,9));
    set(line3_2,'XData',time,'YData',plot_data(:,10));
    set(line3_3,'XData',time,'YData',plot_data(:,11));
%%
    %%%%%--------------------save------------------
    for i=1:28
        if size(OUT_data(i).nodeNo,1) ~= 0
            if save_data.start_swith==1 &
save_data.stop_swith==0 %#ok<AND2>
                if size(OUT_data(i).save_start_time,1)==0

OUT_data(i).save_start_time=OUT_data(i).time_off(end);

OUT_data(i).save_stop_time=OUT_data(i).time_off(end);

OUT_data(i).save_start_ren=size(OUT_data(i).time_off,1);

OUT_data(i).save_stop_ren=size(OUT_data(i).time_off,1);
                else

OUT_data(i).save_stop_time=OUT_data(i).time_off(end);

OUT_data(i).save_stop_ren=size(OUT_data(i).time_off,1);
                end
            elseif save_data.start_swith==0 &
save_data.stop_swith==1 %#ok<AND2>
            else
                OUT_data(i).save_start_time=[];
                OUT_data(i).save_stop_time=[];
                OUT_data(i).save_start_ren=[];
                OUT_data(i).save_stop_ren=[];
            end
        else
            OUT_data(i).save_start_time=[];
            OUT_data(i).save_stop_time=[];
            OUT_data(i).save_start_ren=[];
            OUT_data(i).save_stop_ren=[];
        end
    end

    set(ui_10,'string',OUT_data(nodeNo_plot).save_start_time);
    set(ui_12,'string',OUT_data(nodeNo_plot).save_stop_time);

    save_node_old={};
```

```
save_node_old(1)={'All'};
    node_length=[];
    for i=1:28
        if size(OUT_data(i).nodeNo,1) ~= 0
            k=OUT_data(i).nodeNo;
            kk=length(save_node_old);

eval(['save_node_old(',num2str(kk+1),')={''',num2str(k),'''};']);
            node_length=[node_length,k];
        else
        end
    end
    if length(save_node_old) == 1
        save_data.save_node_old={' '};
        save_data.node_length=[];
    else
        save_data.save_node_old=save_node_old;
        save_data.node_length=node_length;
    end
    set(ui_17,'string',save_data.save_node_old);


%%
function Terminate(block)
    calllib('ImuzMatInterface', 'ImuzIf_Close');
    unloadlibrary('ImuzMatInterface');
    unloadlibrary('ImuzCommunication');
    unloadlibrary('SimpleCom');
```

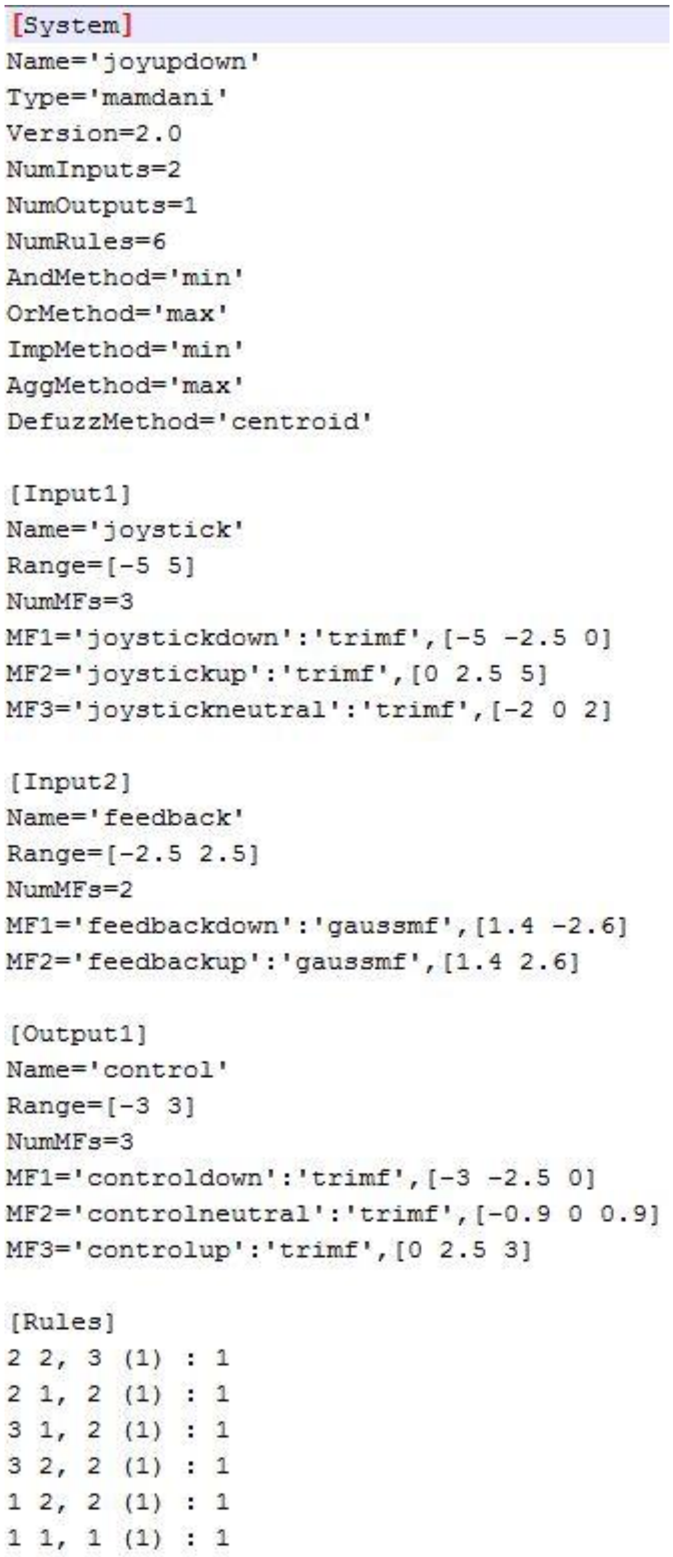
```
% Imuz_CAN_start.m

global OUT_data nodeNo_plot ui_8 ui_4 ui_6  CB_old bode_old

    for i=1:28
        OUT_data(i).nodeNo=[];
        OUT_data(i).time=[];
        OUT_data(i).data=[];
        OUT_data(i).time_off=[];
        OUT_data(i).save_start_time=[];
        OUT_data(i).save_stop_time=[];
        OUT_data(i).save_start_ren=[];
        OUT_data(i).save_stop_ren=[];
    end
    kk1=get(ui_4,'Value');
    kk2=get(ui_6,'Value');
    portName=[char(CB_old(kk1)),'(', char(bode_old(kk2)),')'];

    nodeNo_plot=get(ui_8,'Value');

    set_param(simmodel,'SimulationCommand','start');

    set(ui_3,'Enable','Off');
    set(ui_4,'Enable','Off');
    set(ui_5,'Enable','Off');
    set(ui_6,'Enable','Off');

    set(ui_13,'Enable','ON');
    set(ui_14,'Enable','ON');
    set(ui_15,'Enable','ON');
```

```
% Imuz_CAN_stop.m

set_param(simmodel,'SimulationCommand','stop');

set(ui_3,'Enable','ON');
set(ui_4,'Enable','ON');
set(ui_5,'Enable','ON');
set(ui_6,'Enable','ON');

set(ui_13,'Enable','OFF');
set(ui_14,'Enable','OFF');
set(ui_15,'Enable','OFF');

save_data.start_swith=0;save_data.stop_swith=1;
```

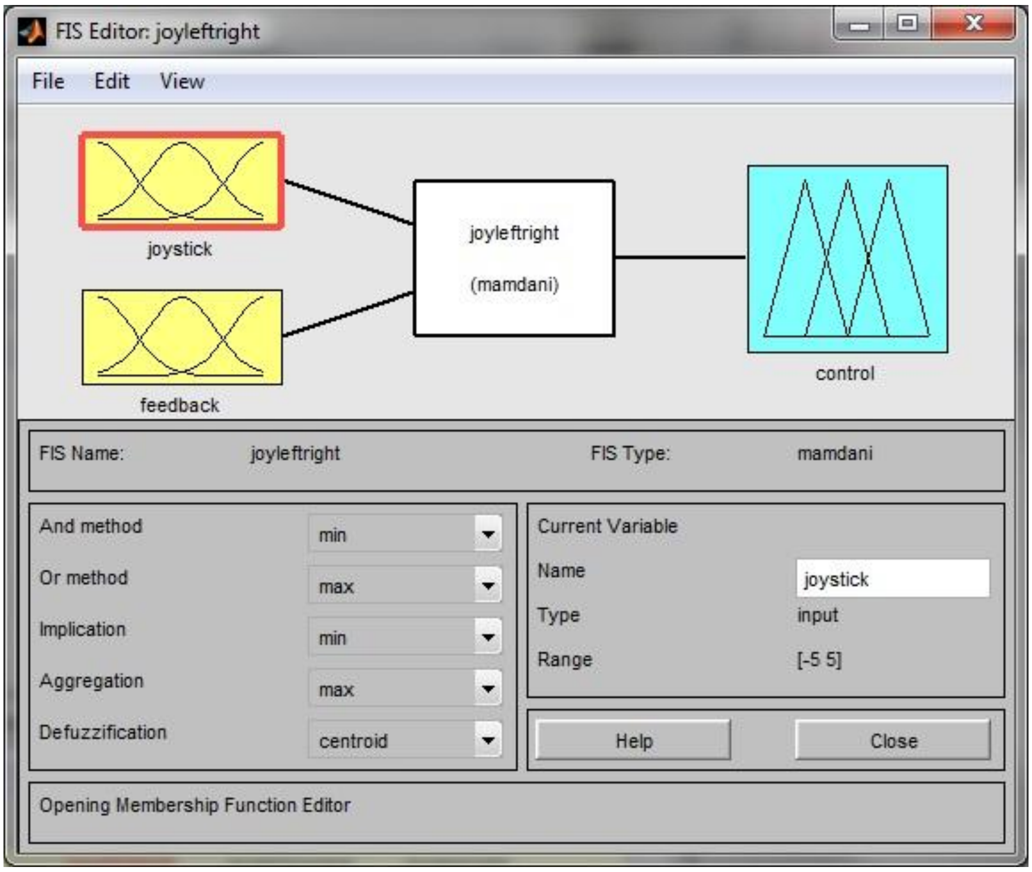
```
% Imuz_data_save.m

global OUT_data save_data ui_17
if save_data.start_swith==0 & save_data.stop_swith==1    %#ok<AND2>
   if save_data.node_length ~= 0 & size(get(ui_10,'string'),1) ~=0 &
size(get(ui_12,'string'),1)~=0 %#ok<AND2>

       save_data_node=get(ui_17,'value');

       title6    = char(save_data.label_box(1));
       format    = '%f';
       for ii = 2:11
           title6    = [title6,' ',char(save_data.label_box(ii))];
           format    = [format '\t%f'];
       end
       format    = [format '\r\n'];

       if save_data_node == 1
           run_data=save_data.node_length;
       else
           run_data=save_data_node-1;
       end

       for i=run_data;
           DataGr_old=OUT_data(i).data;
           DataGr_old(:,2)=OUT_data(i).time_off;

DataGr_old=DataGr_old(OUT_data(i).save_start_ren:OUT_data(i).save_stop
_ren,:);
           fname=['IMUZ_save_data_',num2str(i),'.txt'];
           if length(fname) > 2
               dialogTitle='Save As...';
               cd(save_data.save_path);
               save_path=save_data.save_path;
               [fname,save_path] = uiputfile(fname,dialogTitle);
               if fname == 0
                   break
               end
               cd(save_data.save_path);
               if ~ischar(fname)
                   return;
               end

               fid = fopen(fname,'w');
               fprintf(fid,'%s\r\n',title6);
               fprintf(fid, format, DataGr_old');
               fclose(fid);
               clear DataGr_old i fname save_path
           end
       end
       clear title6 format run_data
   else
   end
else
end
```

APPENDIX 12

FIS-file of forward and backward movement.



```
[System]
Name='joyupdown'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=6
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='joystick'
Range=[-5 5]
NumMFs=3
MF1='joystickdown':'trimf',[-5 -2.5 0]
MF2='joystickup':'trimf',[0 2.5 5]
MF3='joystickneutral':'trimf',[-2 0 2]

[Input2]
Name='feedback'
Range=[-2.5 2.5]
NumMFs=2
MF1='feedbackdown':'gaussmf',[1.4 -2.6]
MF2='feedbackup':'gaussmf',[1.4 2.6]

[Output1]
Name='control'
Range=[-3 3]
NumMFs=3
MF1='controldown':'trimf',[-3 -2.5 0]
MF2='controlneutral':'trimf',[-0.9 0 0.9]
MF3='controlup':'trimf',[0 2.5 3]

[Rules]
2 2, 3 (1) : 1
2 1, 2 (1) : 1
3 1, 2 (1) : 1
3 2, 2 (1) : 1
1 2, 2 (1) : 1
1 1, 1 (1) : 1
```

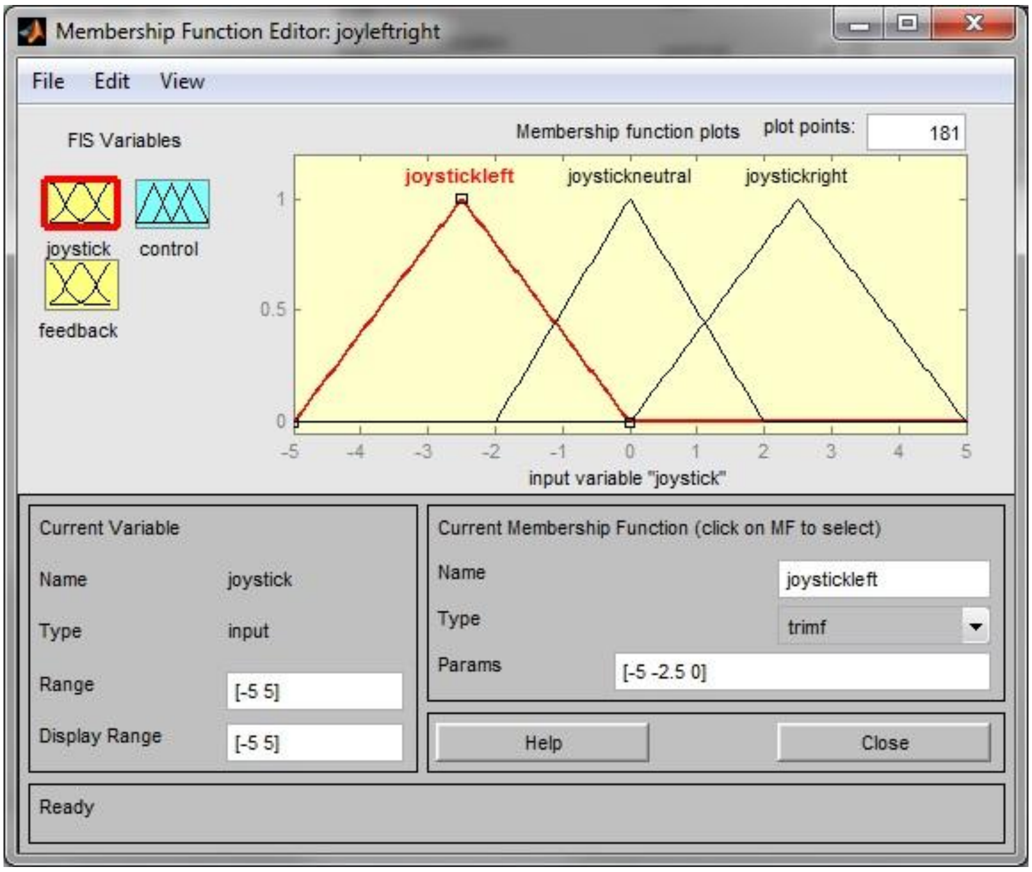


FIS Name: joyleftright
FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: joystick
Type: input
Range: [-5 5]

Opening Membership Function Editor

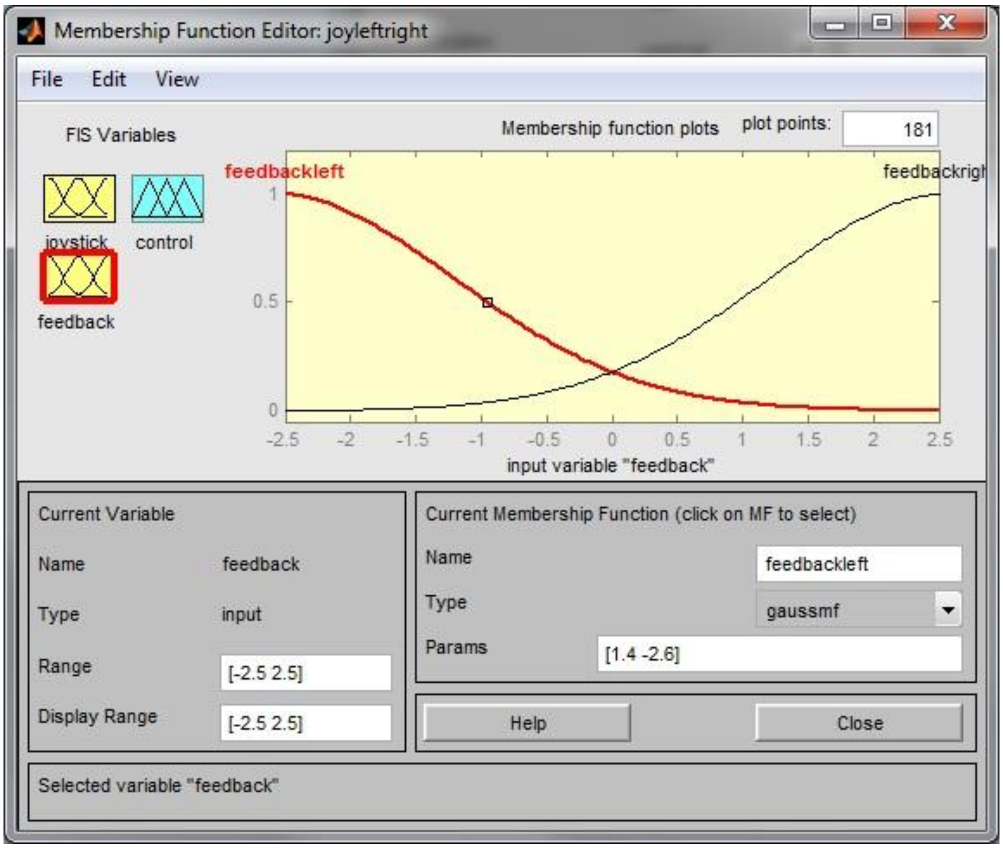


plot points: 181

Current Variable
Name: joystick
Type: input
Range: [-5 5]
Display Range: [-5 5]

Current Membership Function (click on MF to select)
Name: joystickleft
Type: trimf
Params: [-5 -2.5 0]

Ready



plot points: 181

Current Variable

Name feedback

Type input

Range [-2.5 2.5]

Display Range [-2.5 2.5]

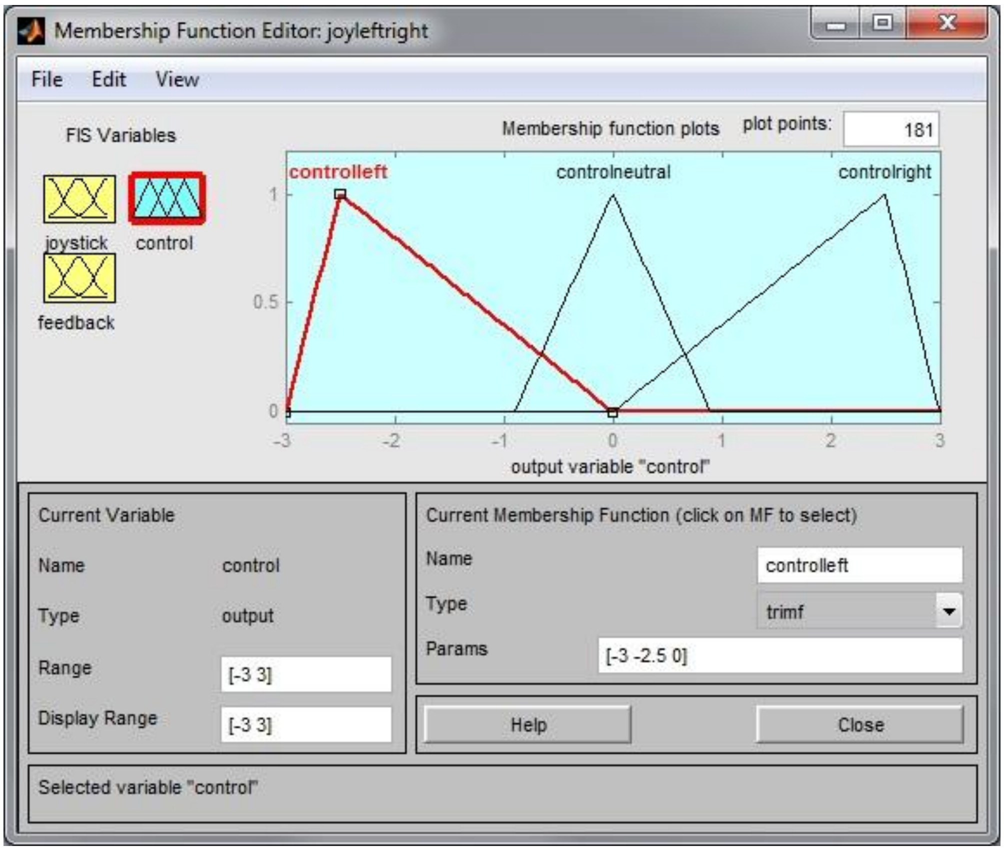
Current Membership Function (click on MF to select)

Name feedbackleft

Type gaussmf

Params [1.4 -2.6]

Selected variable "feedback"



plot points: 181

Current Variable

Name control

Type output

Range [-3 3]

Display Range [-3 3]

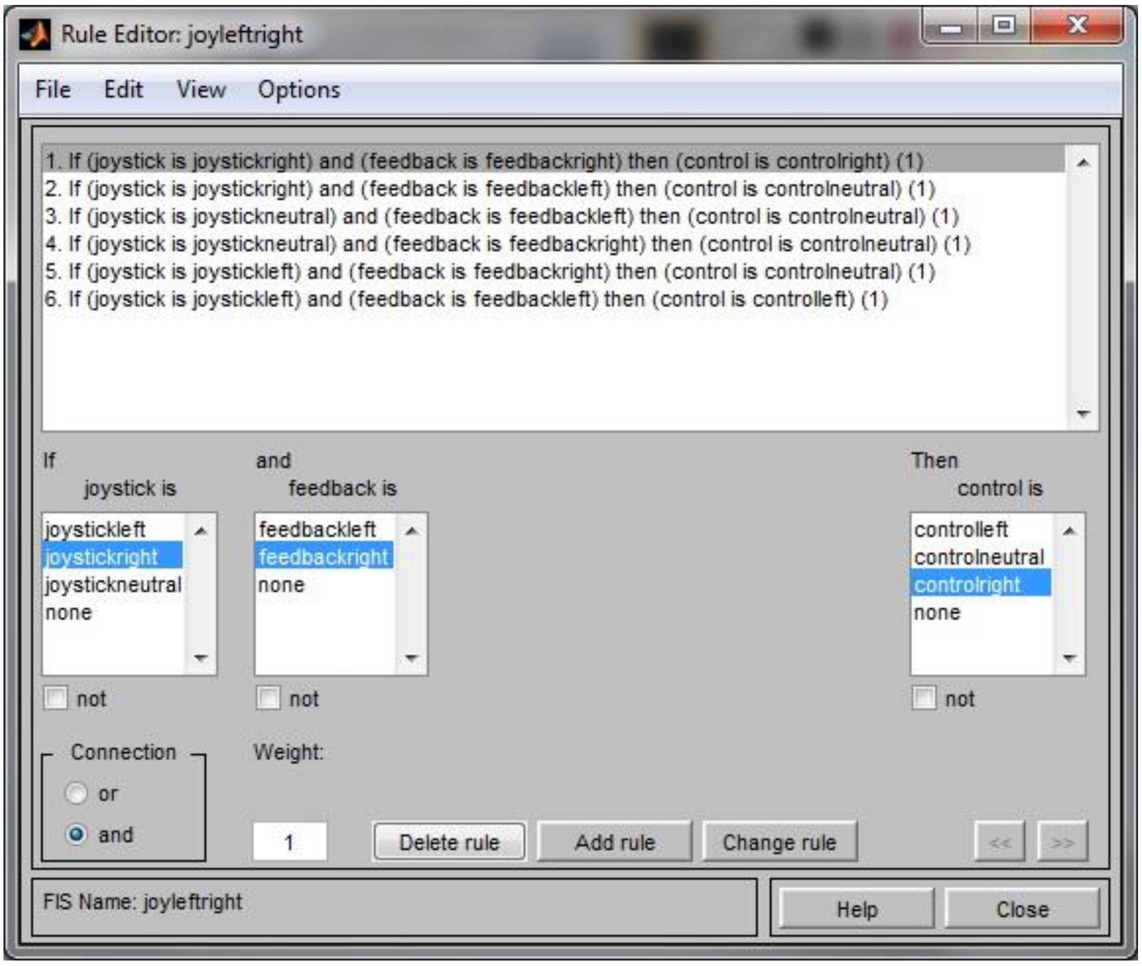
Current Membership Function (click on MF to select)

Name controlleft

Type trimf

Params [-3 -2.5 0]

Selected variable "control"



Rule Editor: joyleftright

1. If (joystick is joystickright) and (feedback is feedbackright) then (control is controlright) (1)
2. If (joystick is joystickright) and (feedback is feedbackleft) then (control is controlneutral) (1)
3. If (joystick is joysticknetural) and (feedback is feedbackleft) then (control is controlneutral) (1)
4. If (joystick is joysticknetural) and (feedback is feedbackright) then (control is controlneutral) (1)
5. If (joystick is joystickleft) and (feedback is feedbackright) then (control is controlneutral) (1)
6. If (joystick is joystickleft) and (feedback is feedbackleft) then (control is controlleft) (1)

If joystick is: joystickleft, [✓ joystickright], joystickneutral, none — not [ ]

and feedback is: feedbackleft, [✓ feedbackright], none — not [ ]

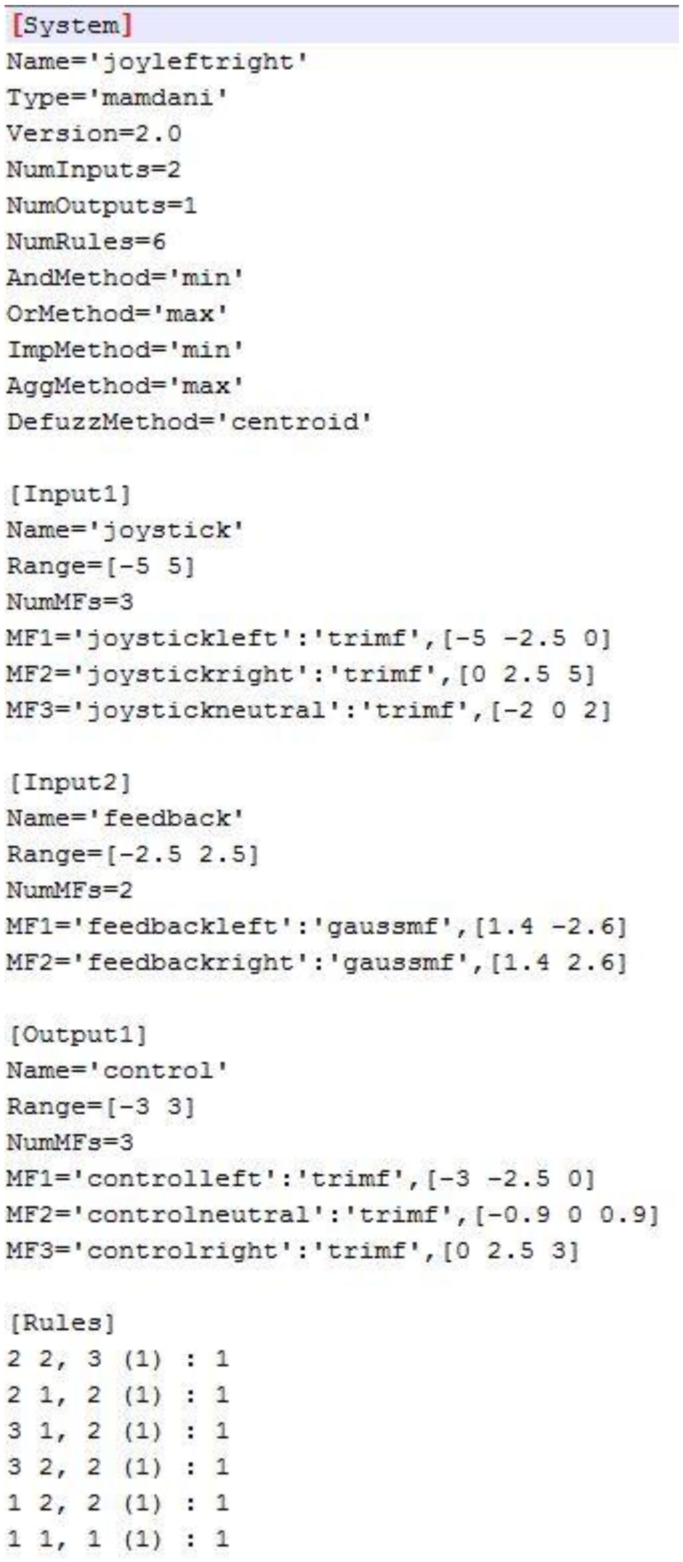
Then control is: controlleft, controlneutral, [✓ controlright], none — not [ ]

Connection: or [✓ and]

Weight: 1

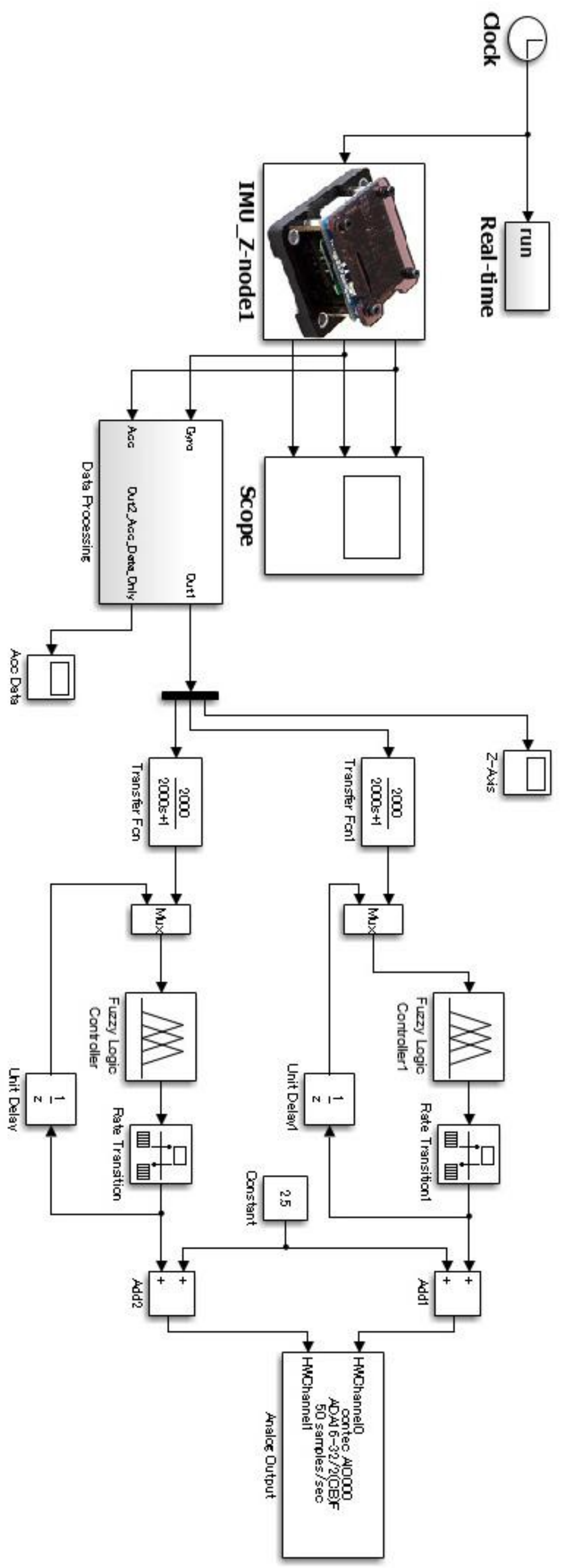
FIS Name: joyleftright

FIS-file of turning movement.



```
[System]
Name='joyleftright'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=6
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='joystick'
Range=[-5 5]
NumMFs=3
MF1='joystickleft':'trimf',[-5 -2.5 0]
MF2='joystickright':'trimf',[0 2.5 5]
MF3='joystickneutral':'trimf',[-2 0 2]

[Input2]
Name='feedback'
Range=[-2.5 2.5]
NumMFs=2
MF1='feedbackleft':'gaussmf',[1.4 -2.6]
MF2='feedbackright':'gaussmf',[1.4 2.6]

[Output1]
Name='control'
Range=[-3 3]
NumMFs=3
MF1='controlleft':'trimf',[-3 -2.5 0]
MF2='controlneutral':'trimf',[-0.9 0 0.9]
MF3='controlright':'trimf',[0 2.5 3]

[Rules]
2 2, 3 (1) : 1
2 1, 2 (1) : 1
3 1, 2 (1) : 1
3 2, 2 (1) : 1
1 2, 2 (1) : 1
1 1, 1 (1) : 1
```

APPENDIX 14


Clock
run
Real-time
IMU_Z-node1
Scope
Gyro
Acc
Out1
Out2_Acc_Data_Only
Data Processing
Acc Data
Z-Axis
2000
2000s+1
Transfer Fcn
2000
2000s+1
Transfer Fcn1
Mux
Mux
Fuzzy Logic Controller
Fuzzy Logic Controller1
1
z
Unit Delay
1
z
Unit Delay1
Rate Transition
Rate Transition1
2.5
Constant
Add2
Add1
HWChannel0
contec AIO000
ADA16-32/2(CB)F
50 samples/sec
HWChannel1
Analog Output