MEP-Pi

User Guide v1.0.1

MINT ROBOT

Table of Contents

Copyright & Trademark Notice

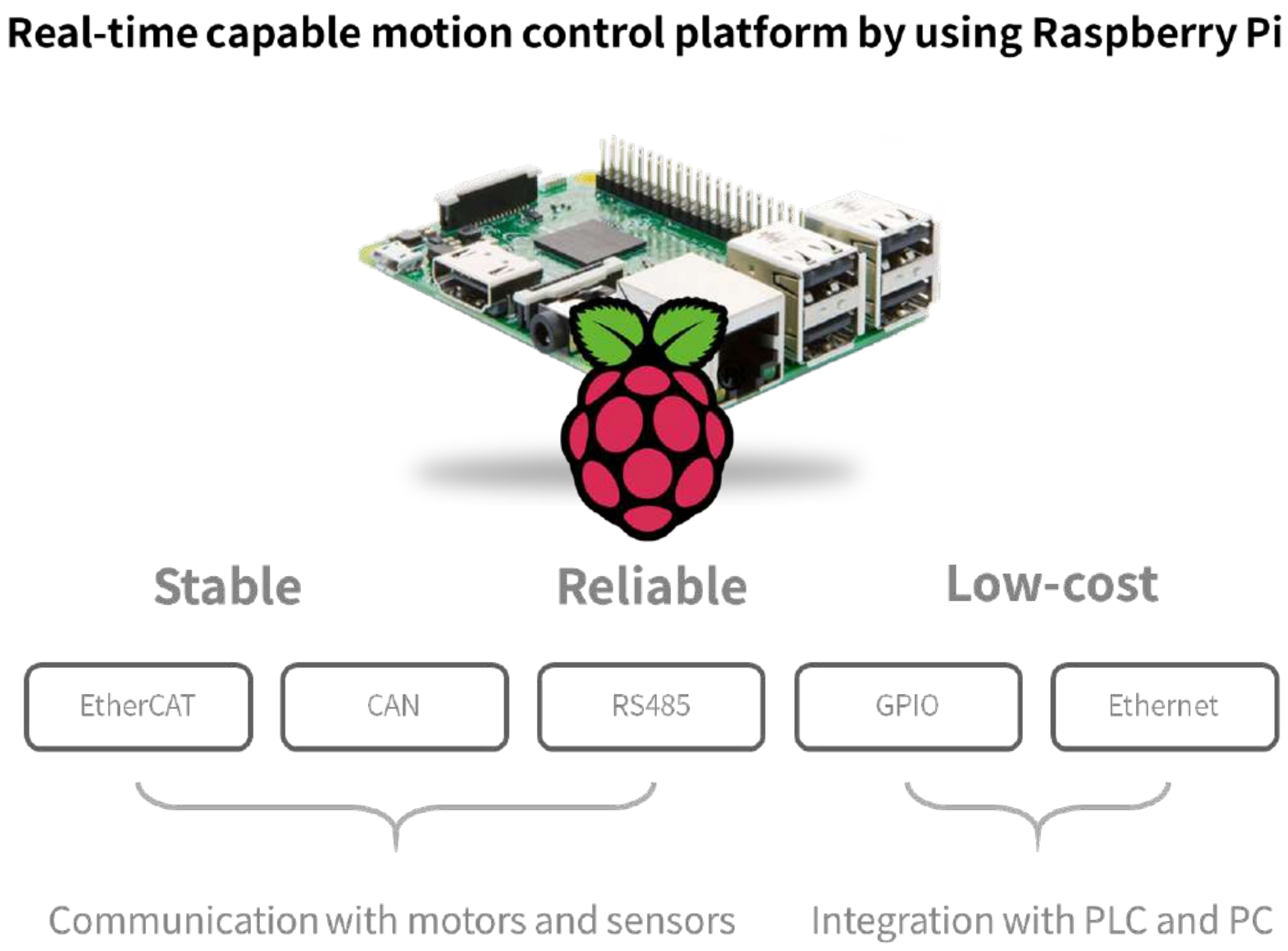
About this guide

This guide is designed to get you up and running with the basic concept and key features of your new MEP-Pi. This guide contains the basic information of the product with the several important precautions. You can study about the basic tutorial in order to implement your own motion control application.

If you have any questions while using MEP-Pi, please contact us by using following email : support@mintrobot.co.kr

1. Introduction to MEP-Pi

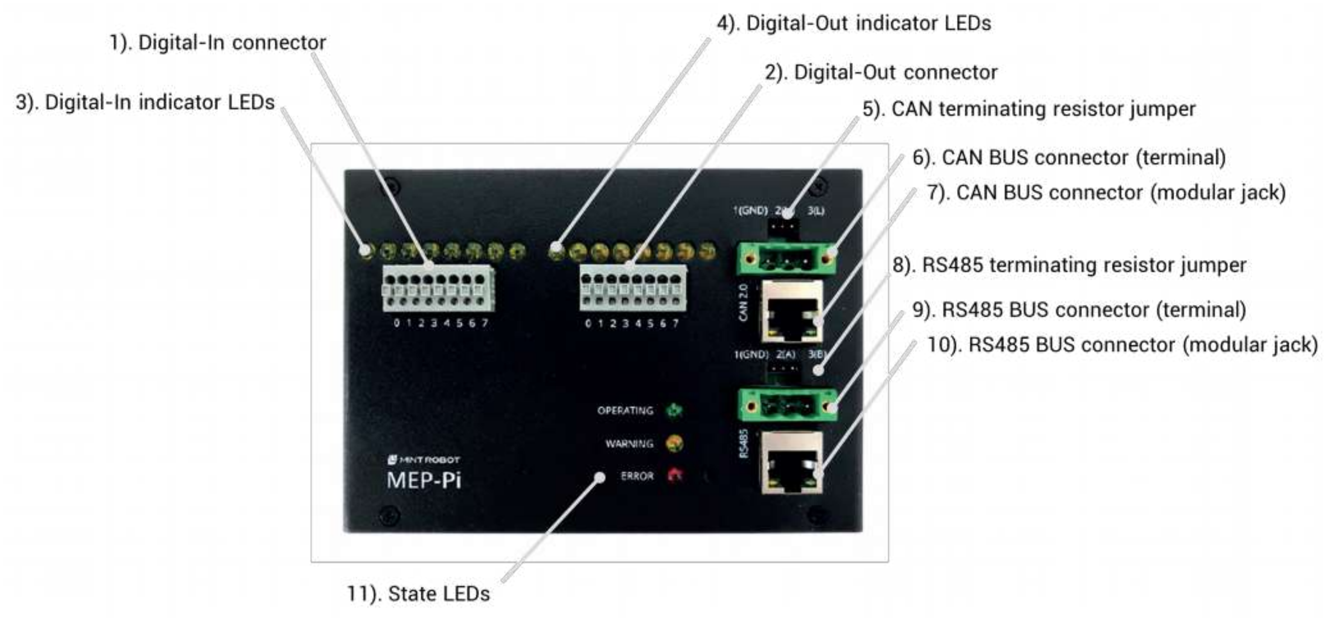
1.1 Basic concept


Real-time capable motion control platform by using Raspberry Pi
Stable
Reliable
Low-cost
EtherCAT
CAN
RS485
GPIO
Ethernet
Communication with motors and sensors
Integration with PLC and PC

<Figure 1> Concept of MEP-Pi

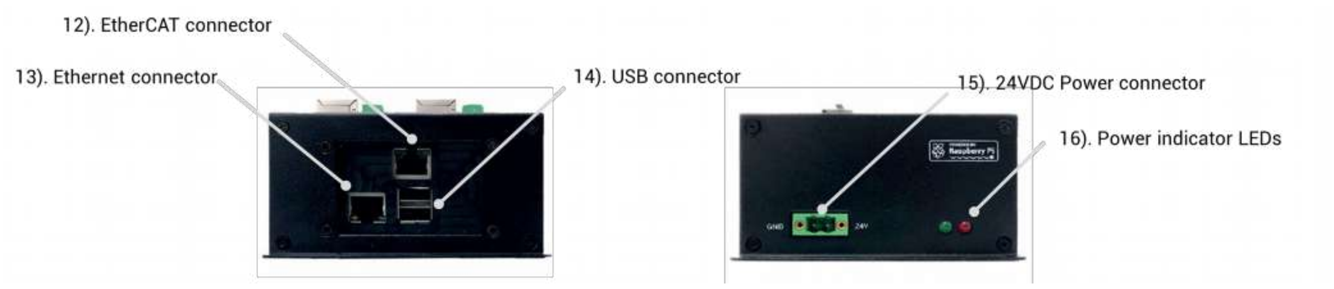
MEP-Pi is a low-cost motion control platform by using Raspberry Pi. It designed to substitute existing expensive motion controllers with more cost-effective solution. It is stable because it is based on Raspbian operating system which is already verified by many users. It is real-time capable to implement reliable discrete feedback control system. User can communicate with motors and sensors by using fieldbus communication such as EtherCAT, CAN, and RS485. Also, users can communicate with higher control system such as PLC and PC.

1.2 Appearance


1). Digital-In connector
3). Digital-In indicator LEDs
4). Digital-Out indicator LEDs
2). Digital-Out connector
5). CAN terminating resistor jumper
6). CAN BUS connector (terminal)
7). CAN BUS connector (modular jack)
8). RS485 terminating resistor jumper
9). RS485 BUS connector (terminal)
10). RS485 BUS connector (modular jack)
11). State LEDs
MEP-Pi
OPERATING
WARNING
ERROR

<Figure 2> Top view of MEP-Pi

1). Digital-In connector	24V DC digital input signal connector
2). Digital-Out connector	24V DC digital output signal connector
3). Digital-In indicator LEDs	LEDs of each port of digital input signals
4). Digital-Out indicator LEDs	LEDs of each port of digital output signals
5). CAN terminating resistor jumper	Jumper to activate terminating resisteor for CAN interface
6). CAN BUS connector (terminal)	Terminal connector of CAN interface
7). CAN BUS connector (modular jack)	Modular jack connector of CAN interface
8). RS485 BUS terminating resistor jumper	Jumper to activate terminating resistor for RS485 interface
9). RS485 BUS connector (terminal)	Terminal connector of RS485 interface
10). RS485 BUS connector (modular jack)	Modular jack connector of RS485 interface
11). State LEDs	LEDs to indicate operating state



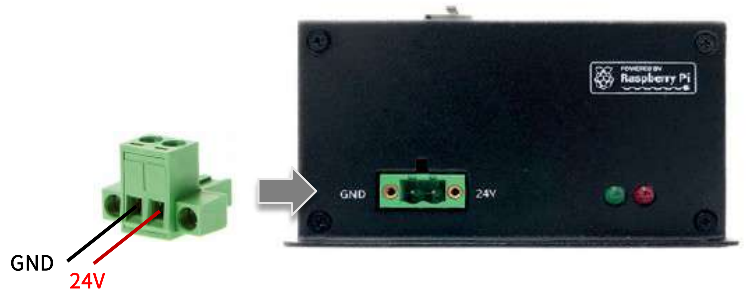
<Figure 3> Left and right view of MEP-Pi

12). EtherCAT connector	RJ45 connector for EtherCAT interface

13). Ethernet connector	RJ45 connector for Ethernet interface
14). USB connectors	2 x USB2.0 connectors for keyboard, mouth, and etc.
15). 24V DC Power connector (Terminal)	Main power connector
16). Power indicator LEDs	LEDs to indicate the status of power source

1.3 Power system

1.3.1 Main Power



LED	Meaning
Green	Applied power is correct
Red	Applied power is inverted

WARNING

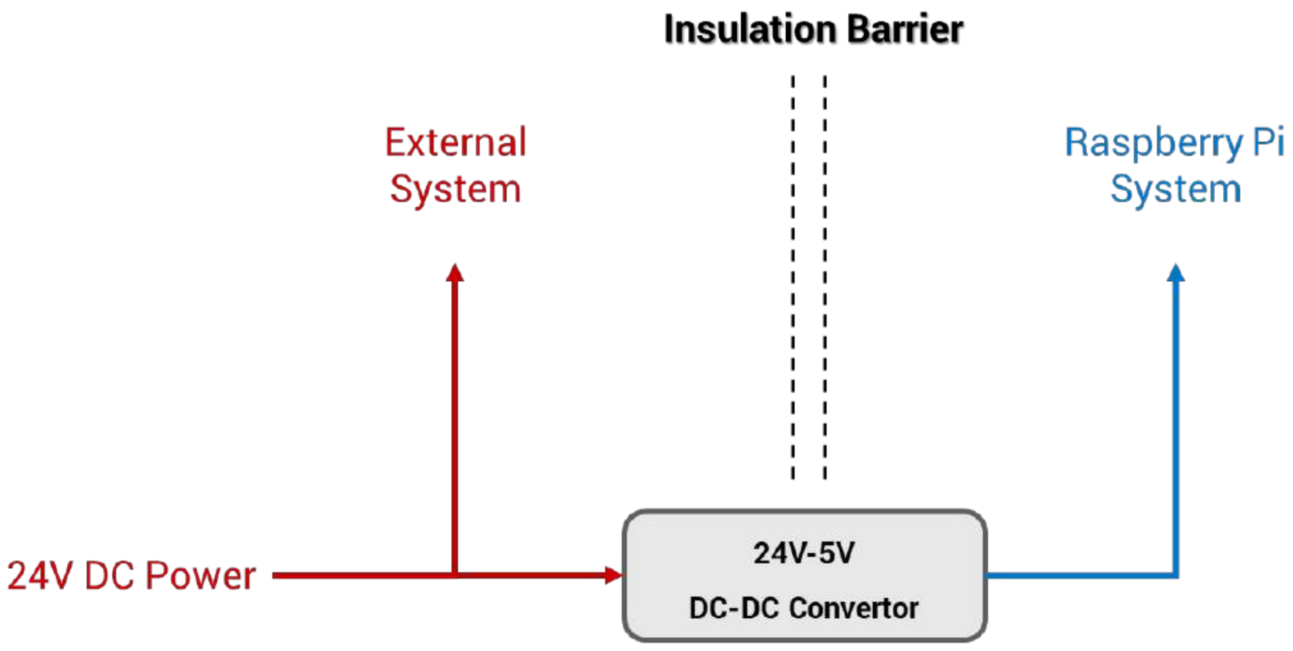
DO NOT EXCEED 24V FOR MAIN POWER

<Figure 4> Main power of MEP-Pi

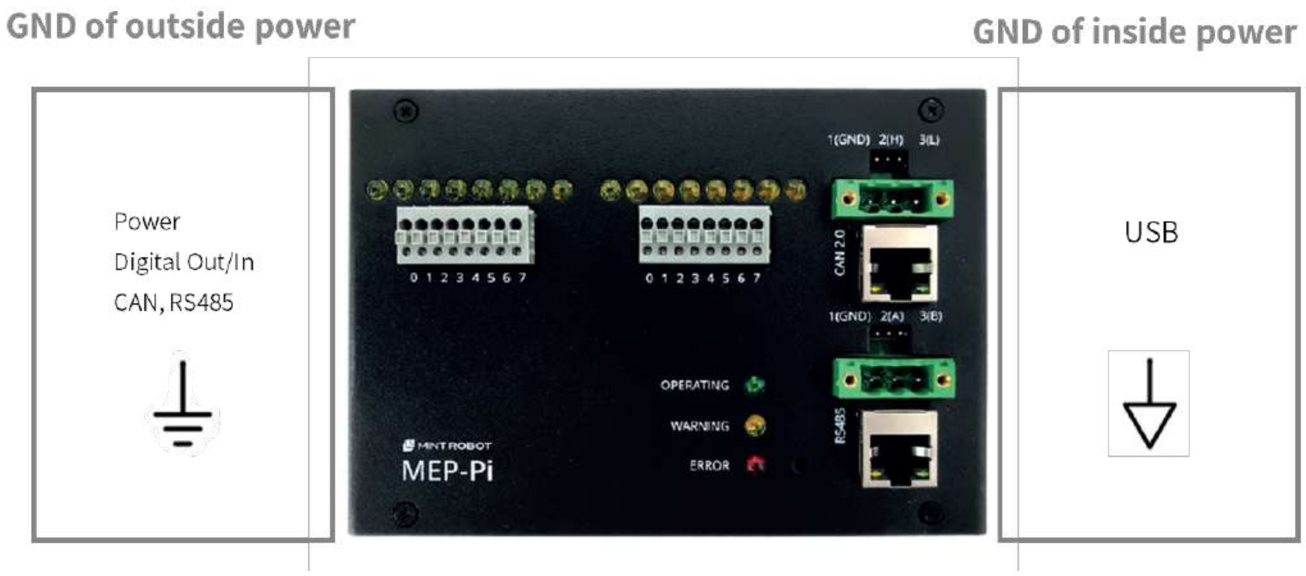
The main power of MEP-Pi is 24V DC which is commonly used for industrial equipment. MEP-Pi turns on automatically without switch if 24V DC power is applied. **NOTE, check the voltage level and check the direction of polarity before applying power.** It is useful when MEP-Pi is integrated with higher system, so it needs to turn on immediately after turning on main power of the system.

1.3.2 Power system schema

MEP-Pi has 24V-5V DC-DC converter which can make isolated 5V voltage power source for internal system including Raspberry Pi. It means the grounds of 24V and 5V are different, so there is no electrical relation between two power sources.



<Figure 5> Insulation of 2 power source of MEP-Pi

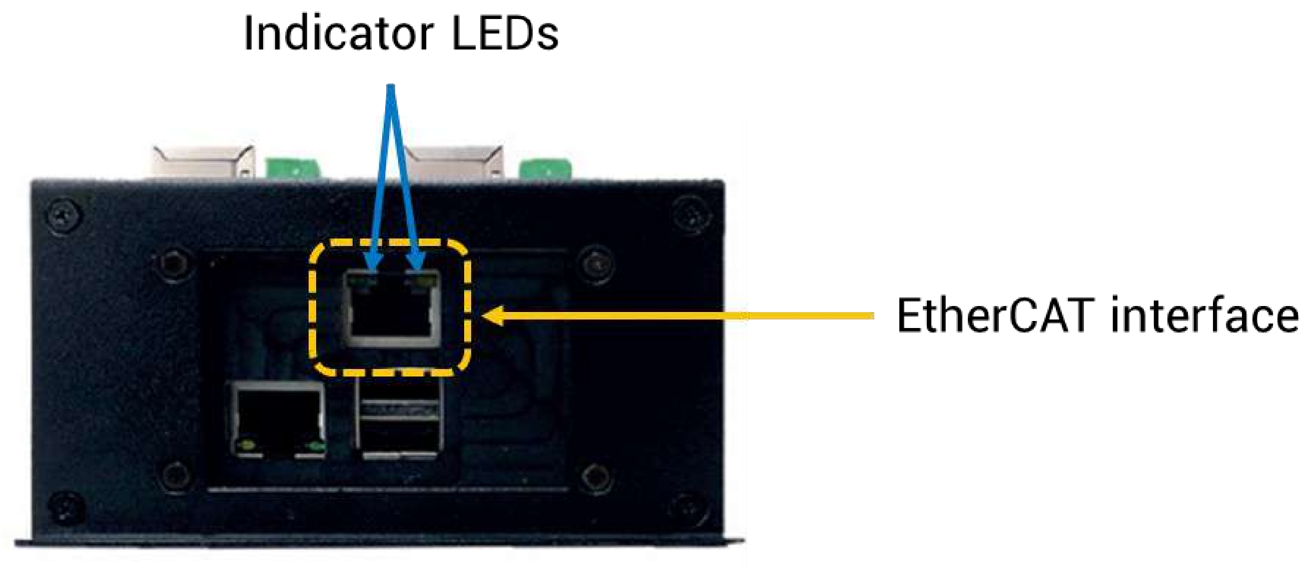


<Figure 6> Insulated ground for the interfaces of MEP-Pi

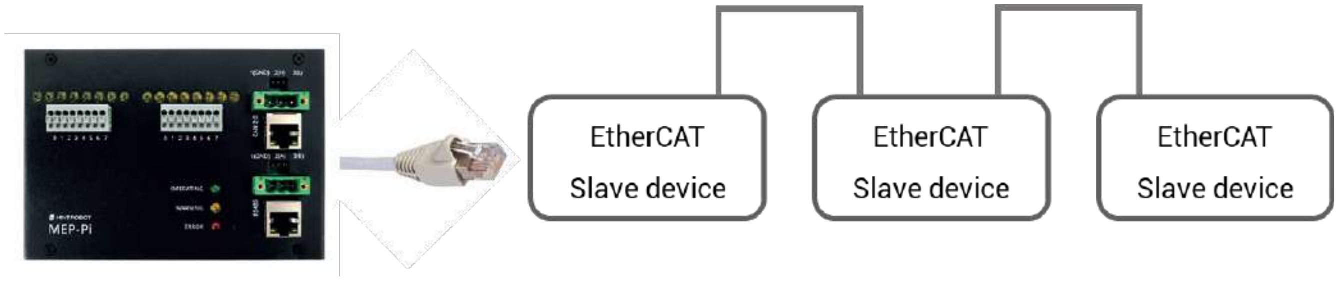
Only USB interface has different ground with other interfaces, which means the ground in USB interface shouldn't be mixed with other interfaces' ground. **NOTE, do not use USB power to operate the devices using external power.** We do not recommend using USB power to operate other device. It would be better to use USB power only for keyboard and mouse.

1.4 Communication interfaces

1.4.1 EtherCAT



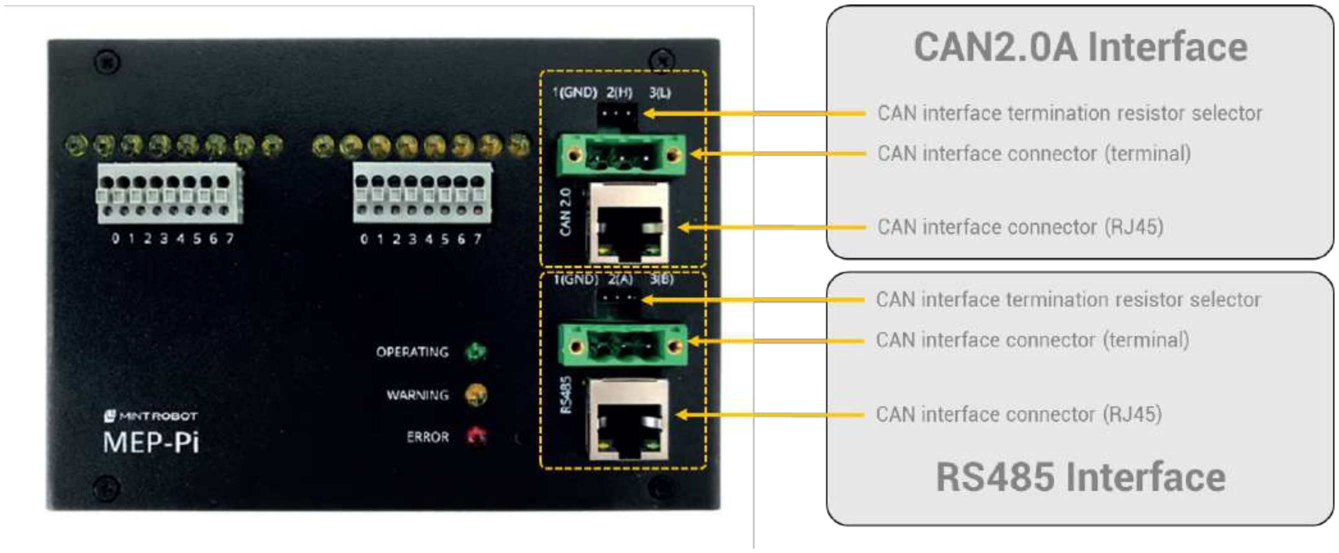
<Figure 7> EtherCAT connector of MEP-Pi



<Figure 8> EtherCAT network by using MEP-Pi

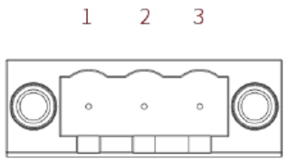
MEP-Pi has 1 x EtherCAT interface to control EtherCAT slave devices. EtherCAT interface should be connected to the input of EtherCAT slave devices by using RJ45 cable. **NOTE, the usage of over CAT5E grade SFTP cable (Shielded and Foiled Twisted Pair) with shield RJ45 connector is recommended.** Once EtherCAT devices are connected to MEP-Pi, the LEDs of EtherCAT connector are blinking, which means MEP-Pi is communicating with the EtherCAT slave devices correctly.

1.4.2 CAN & RS485


CAN2.0A Interface
CAN interface termination resistor selector
CAN interface connector (terminal)
CAN interface connector (RJ45)
CAN interface termination resistor selector
CAN interface connector (terminal)
CAN interface connector (RJ45)
RS485 Interface
1(GND) 2(H) 3(L)
1(GND) 2(A) 3(B)
CAN 2.0
RS485
OPERATING
WARNING
ERROR
MINTROBOT
MEP-Pi
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7

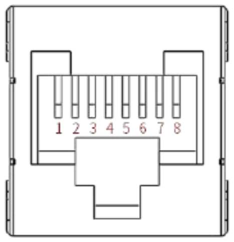
<Figure 9> CAN and RS485 interfaces of MEP-Pi

MEP-Pi has 1 x CAN2.0A and 1 x RS485 interfaces. Each interface uses same connectors. User can select one of the terminal connector and RJ45 connector to connect communication cable. The communication cable must follow the pinout of each interfaces.



Terminal connector pinout

1	GND
2	CAN_H / RS485_A
3	CAN_L / RS485_B

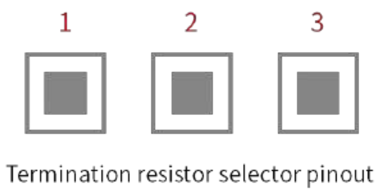


RJ45 connector pinout

1	NC
2	NC
3	CAN_H / RS485_A
4	NC
5	NC
6	CAN_L / RS485_B
7	GND
8	GND

<Figure 10> Pinout of CAN(RS485) connectors

The grounds of each connectors are used only for indicating LEDs of RJ45 connectors. **NOTE, the LEDs of the RJ45 connector are powered by 24V main power source, so the ground of each connector should be same to the ground of the power source to turn on LEDs.** It is not mandatory, but we recommend match the grounds of CAN and RS485 connector to the ground of main power source.

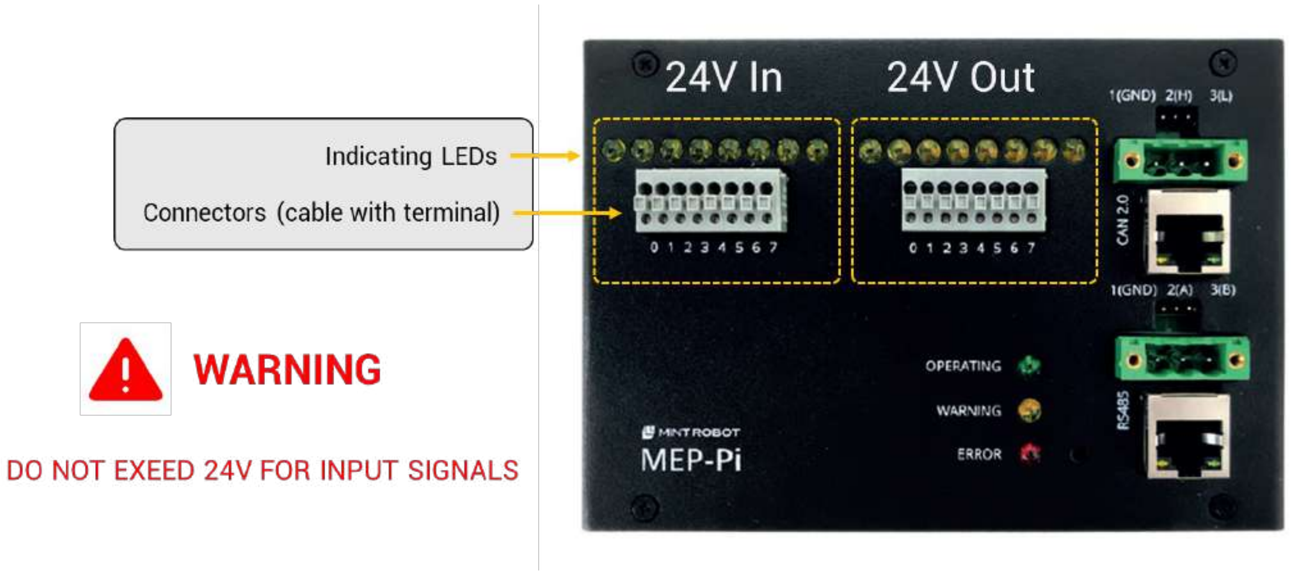


Pin number		Termination resistor
1-2	2-3	
Disconnect	Disconnect	Deactivate
Short	Disconnect	Deactivate
Disconnect	Short	Activate

<Figure 11> terminating resistor selecting pins

Termination resistors of each interface are activated/deactivated by jumping terminating resistor selection pins.

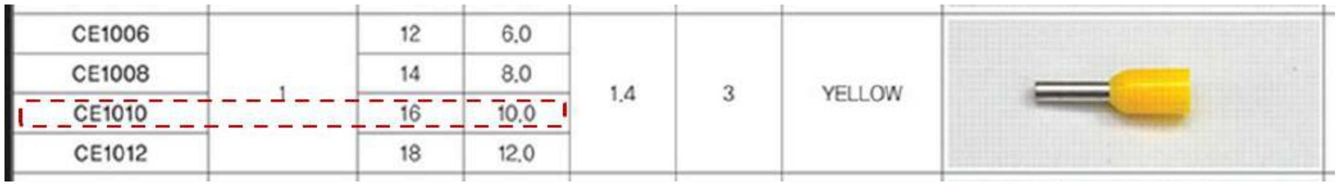
1.4.3 Digital In/Out



<Figure 11> Digital In/Out interfaces of MEP-Pi

MEP-Pi has 8 x 24VDC digital input and 8 x 24VDC digital output interfaces. Each interface has LEDs to indicate state of each port. **NOTE, the voltage to input connector must not**

exceed 24V. The input signals are transmitted by using a photo coupler for the insulation between outside and outside of the system. The photo coupler is powered by the main source of the MEP-Pi, which means the ground of the input signal must be same to the ground of the main power source in order to trigger photo coupler. The same principle is applied for the output signals. **NOTE, The ground of the input/output signals must be same to the ground of power source.**



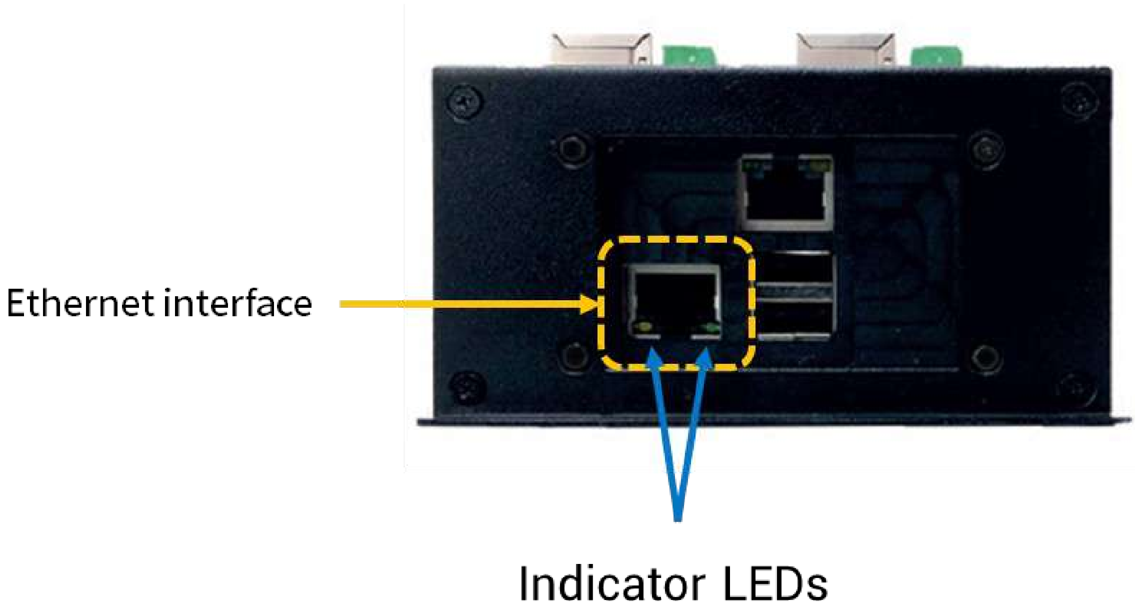
CE1006	1	12	6.0	1.4	3	YELLOW	
CE1008		14	8.0				
CE1010		16	10.0				
CE1012		18	12.0				

* Recommended pen hole terminal is yellow-10mm (wire range 1.0 mm^2)

<Figure 12> Specification of the recommended pin hole terminal

User can use pure wire without terminals, but it is recommended to use pin hole terminals for stable fixing of signal lines.

1.4.4 Ethernet



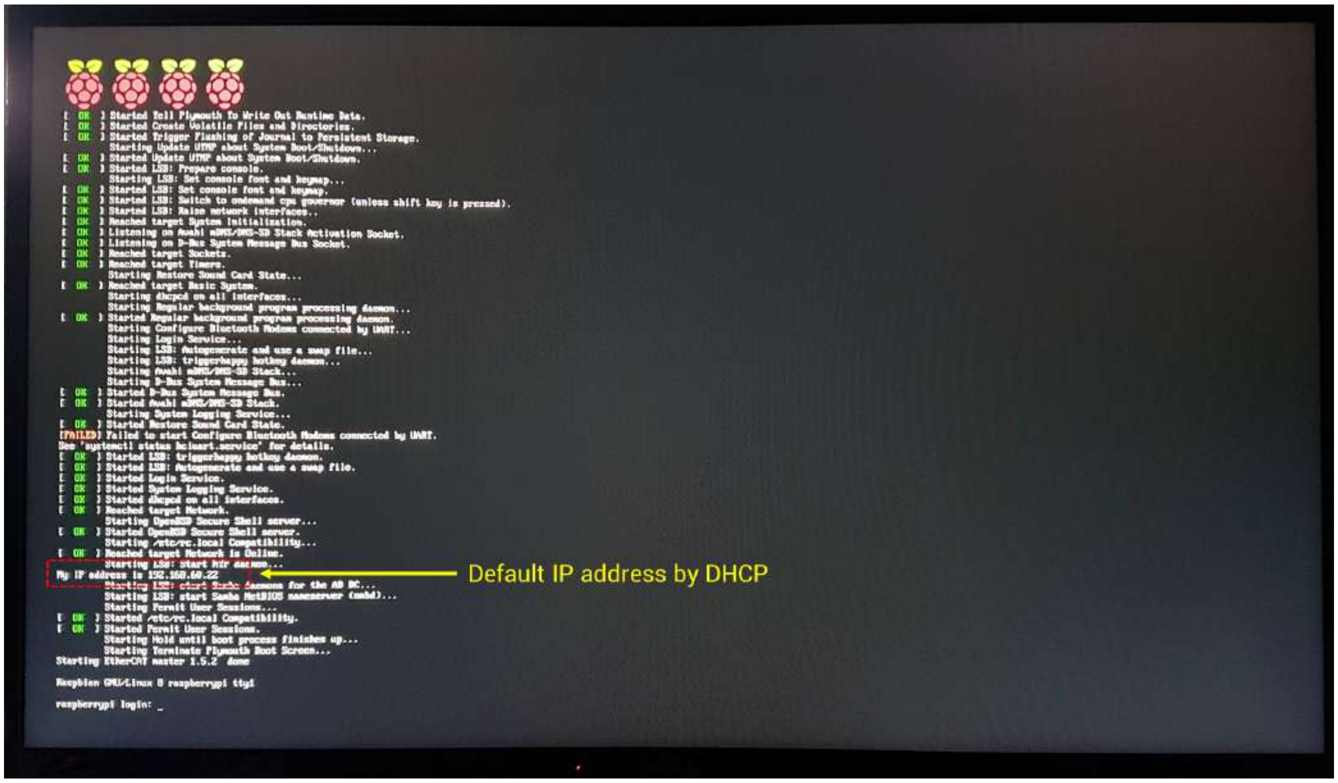
<Figure 13> Ethernet interface of MEP-Pi

MEP-Pi has 1 x Ethernet interface to communicate with other system. **NOTE, the default IP address is DHCP, so it is not fixed at the first time**. User can change IP address by editing interface file at “/etc/network/interface”. We will discuss about this at chapter 2.

2. First connection to MEP-Pi

2.1 Direct connection

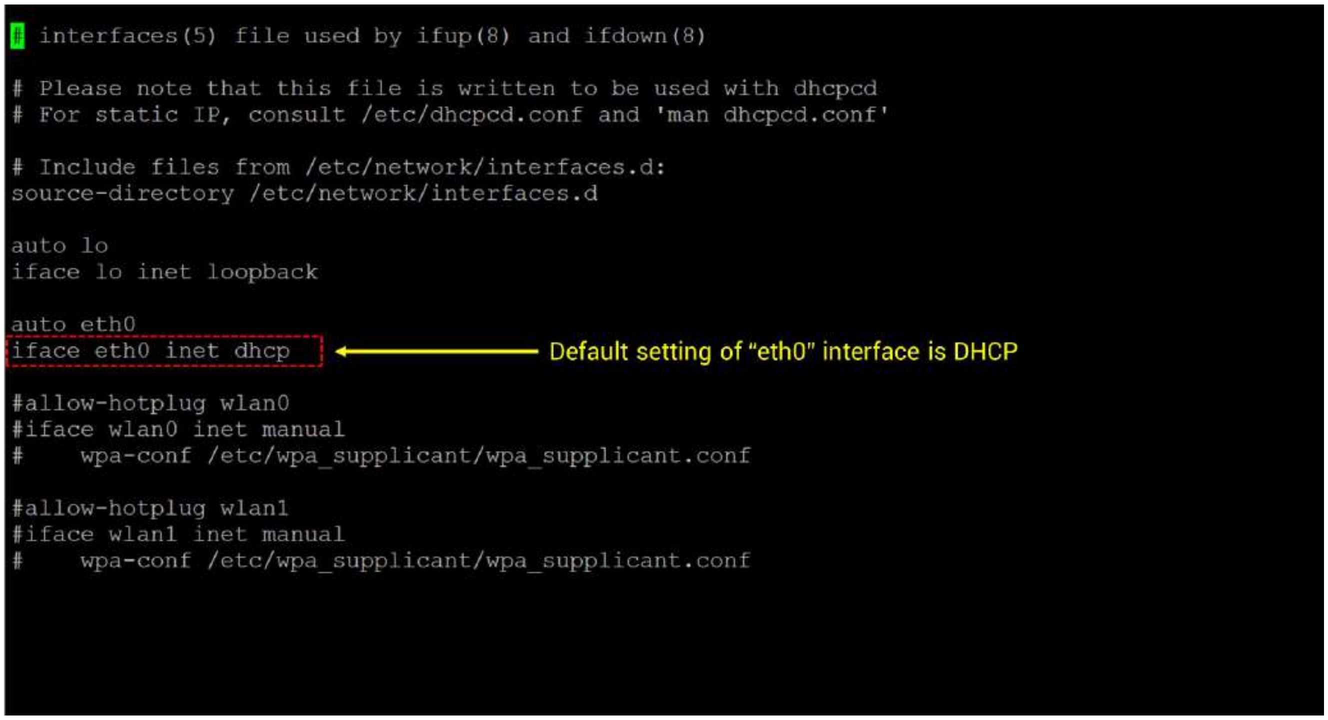
Users need to connect to MEP-Pi directly at the first time in order to set the IP address for the next remote connection. Connect monitor and keyboard to MEP-Pi, then power up MEP-Pi. Then, the booting screen will show up on the monitor.



```
[  OK  ] Started Tell Plymouth To Write Out Runtime Data.
[  OK  ] Started Create Volatile Files and Directories.
[  OK  ] Started Trigger Flushing of Journal to Persistent Storage.
         Starting Update UTMP about System Boot/Shutdown...
[  OK  ] Started Update UTMP about System Boot/Shutdown.
[  OK  ] Started LSB: Prepare console.
         Starting LSB: Set console font and keymap...
[  OK  ] Started LSB: Set console font and keymap.
[  OK  ] Started LSB: Switch to ondemand cpu governor (unless shift key is pressed).
[  OK  ] Started LSB: Raise network interfaces..
[  OK  ] Reached target System Initialization.
[  OK  ] Listening on Avahi mDNS/DNS-SD Stack Activation Socket.
[  OK  ] Listening on D-Bus System Message Bus Socket.
[  OK  ] Reached target Sockets.
[  OK  ] Reached target Timers.
         Starting Restore Sound Card State...
[  OK  ] Reached target Basic System.
         Starting dhcpcd on all interfaces...
         Starting Regular background program processing daemon...
[  OK  ] Started Regular background program processing daemon.
         Starting Configure Bluetooth Modems connected by UART...
         Starting Login Service...
         Starting LSB: Autogenerate and use a swap file...
         Starting LSB: triggerhappy hotkey daemon...
         Starting Avahi mDNS/DNS-SD Stack...
         Starting D-Bus System Message Bus...
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Started Avahi mDNS/DNS-SD Stack.
         Starting System Logging Service...
[  OK  ] Started Restore Sound Card State.
[FAILED] Failed to start Configure Bluetooth Modems connected by UART.
See 'systemctl status hciuart.service' for details.
[  OK  ] Started LSB: triggerhappy hotkey daemon.
[  OK  ] Started LSB: Autogenerate and use a swap file.
[  OK  ] Started Login Service.
[  OK  ] Started System Logging Service.
[  OK  ] Started dhcpcd on all interfaces.
[  OK  ] Reached target Network.
         Starting OpenBSD Secure Shell server...
[  OK  ] Started OpenBSD Secure Shell server.
         Starting /etc/rc.local Compatibility...
[  OK  ] Reached target Network is Online.
         Starting LSB: start Mir daemon...
My IP address is 192.168.60.22
         Starting LSB: start Samba daemons for the AD DC...
         Starting LSB: start Samba NetBIOS nameserver (nmbd)...
         Starting Permit User Sessions...
[  OK  ] Started /etc/rc.local Compatibility.
[  OK  ] Started Permit User Sessions.
         Starting Hold until boot process finishes up...
         Starting Terminate Plymouth Boot Screen...
Starting EtherCAT master 1.5.2  done

Raspbian GNU/Linux 8 raspberrypi tty1

raspberrypi login: _
```

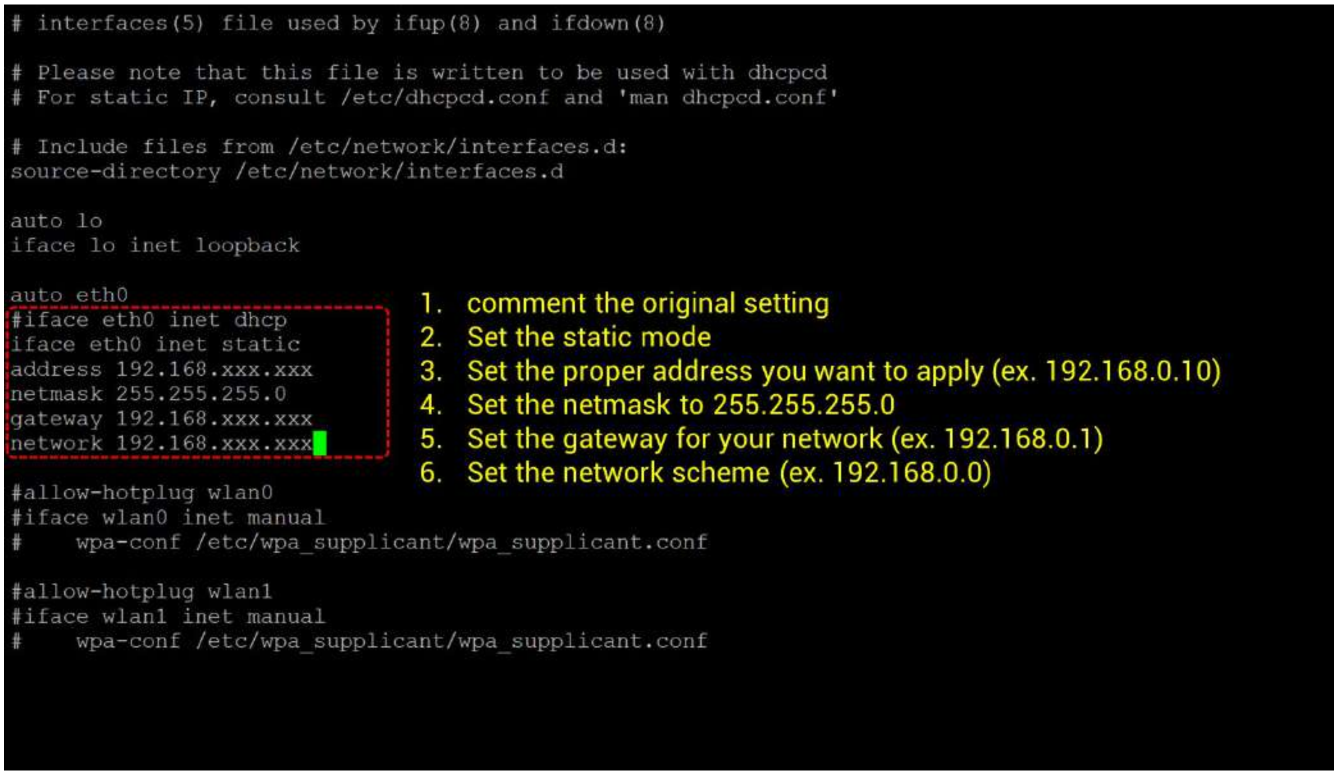
<Figure 14> Boot screen of MEP-Pi

If LAN cable is connected to MEP-Pi, it will have the IP address by DHCP protocol. Users can connect to MEP-Pi remotely by using given IP address. If users want static IP address, some settings should be modified. First, login to MEP-Pi. **NOTE, the default ID and password is "pi / raspberry".** Next, type "sudo nano /etc/network/interfaces". Then, the contents of network setting file will show up. The default setting of "eth0" is dhcp as shown in fig. 15. Users should change this in order to assign static IP address as like fig. 16. Reboot after saving the changed file. Then, the IP address of MEP-Pi is changed.



```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

#allow-hotplug wlan0
#iface wlan0 inet manual
#    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

#allow-hotplug wlan1
#iface wlan1 inet manual
#    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

<Figure 15> Default setting of “interfaces”



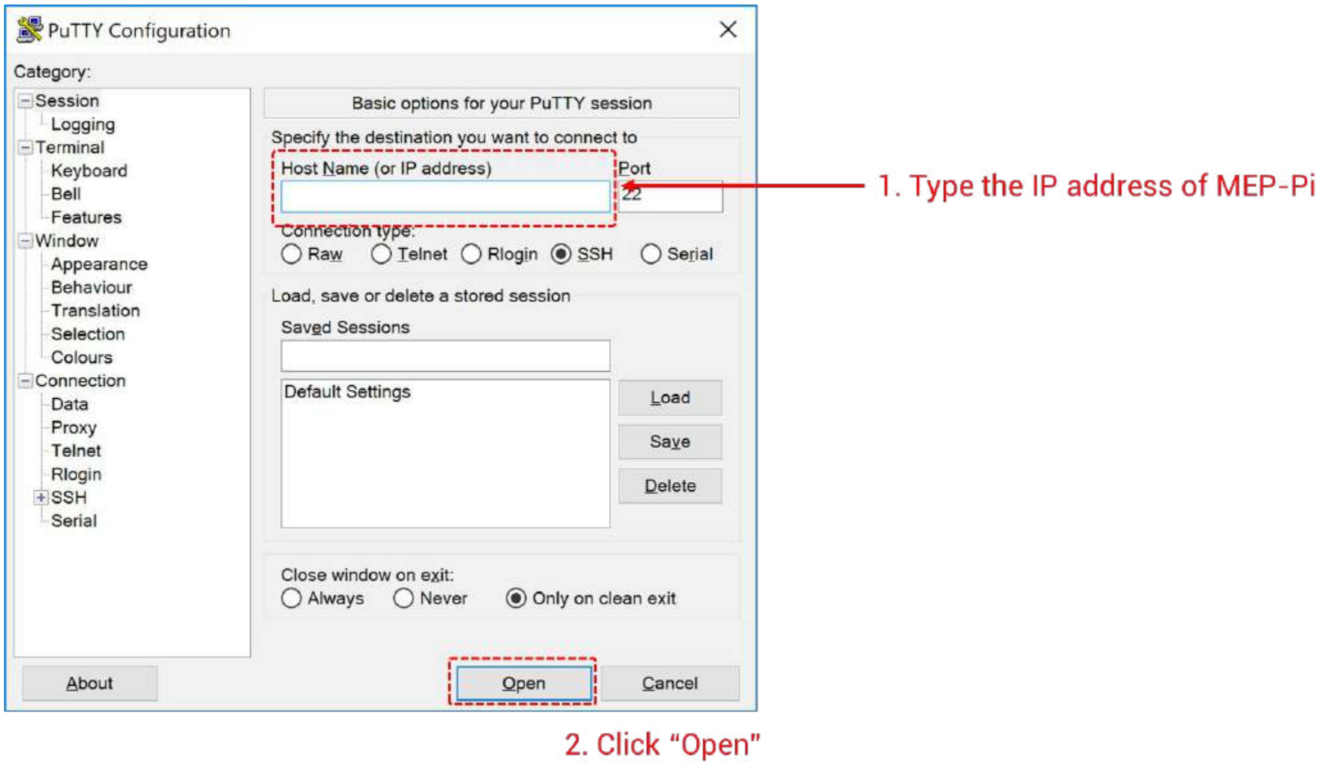
```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
address 192.168.xxx.xxx
netmask 255.255.255.0
gateway 192.168.xxx.xxx
network 192.168.xxx.xxx

#allow-hotplug wlan0
#iface wlan0 inet manual
#    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

#allow-hotplug wlan1
#iface wlan1 inet manual
#    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

1. comment the original setting
2. Set the static mode
3. Set the proper address you want to apply (ex. 192.168.0.10)
4. Set the netmask to 255.255.255.0
5. Set the gateway for your network (ex. 192.168.0.1)
6. Set the network scheme (ex. 192.168.0.0)

<Figure 16> Modified setting of “interfaces” for static IP address

2.2 Remote connection

Users can connect to MEP-Pi by using network. NOTE, users should know the IP address of MEP-Pi before connection. Run “putty.exe” and type the IP address of MEP-Pi. If the connection state is proper, the login screen will show up. Users can login by using same ID and password to the direct connection.



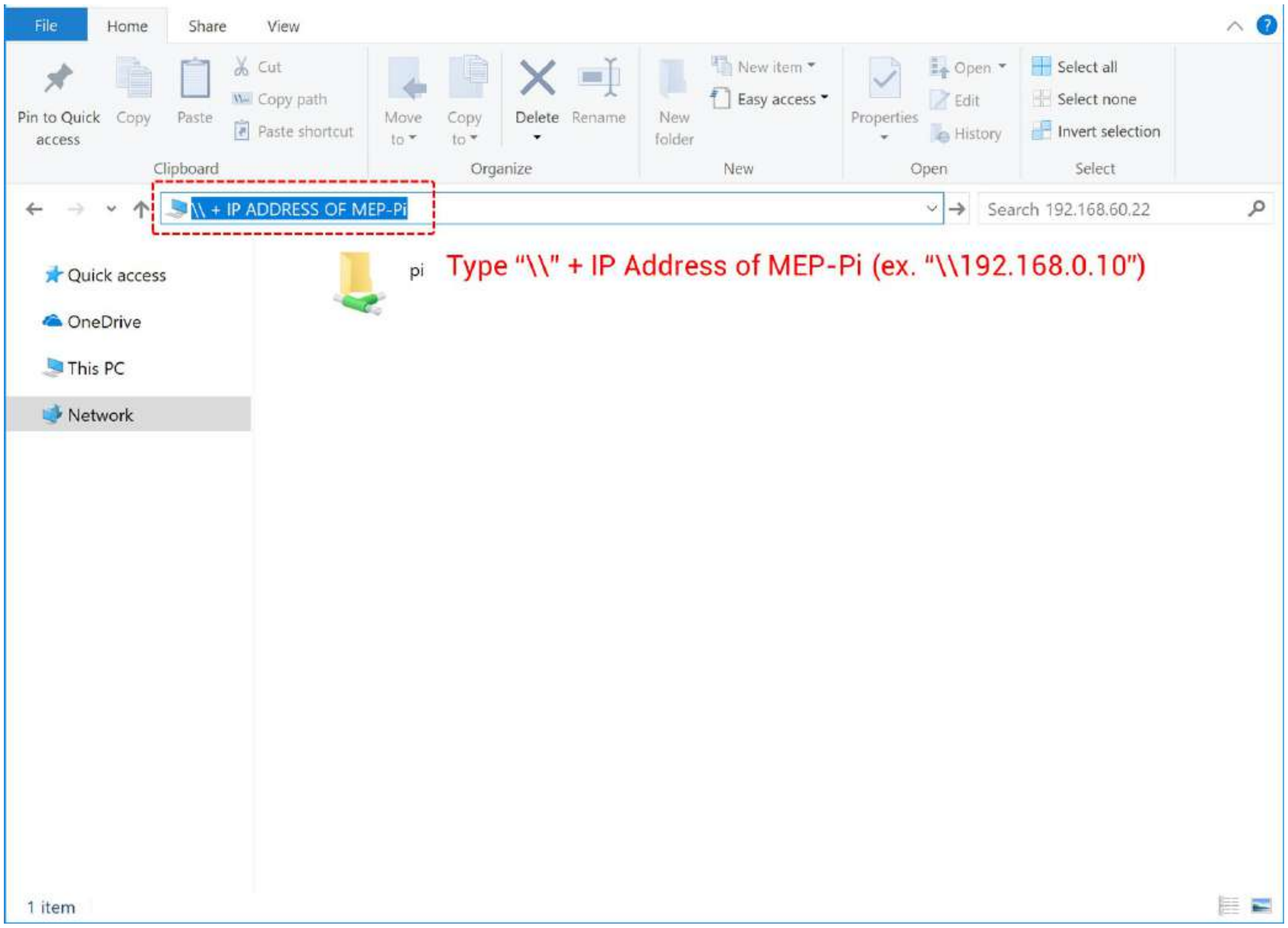
Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial

Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]

<Figure 17> Putty screen for the remote connection

2.3 File transmission

MEP-Pi supports SAMBA server for convenient file transmission. Run “File explorer” and type the IP address of MEP-Pi as like shown in fig. 18. Then, you can see the “pi” folder on y our file explorer. It is “/home/pi” folder in MEP-Pi. Click the folder icon to enter the folder. Then, the login screen will show up. Type “pi / raspberry” for the ID and password, respectively. **NOTE, the ID and password for the samba server login is same to the ID and password for the login but it is differently used, which means even though the ID and password for the login to MEP-Pi is changed, the ID and password for the login to SAMBA server would not be changed.**



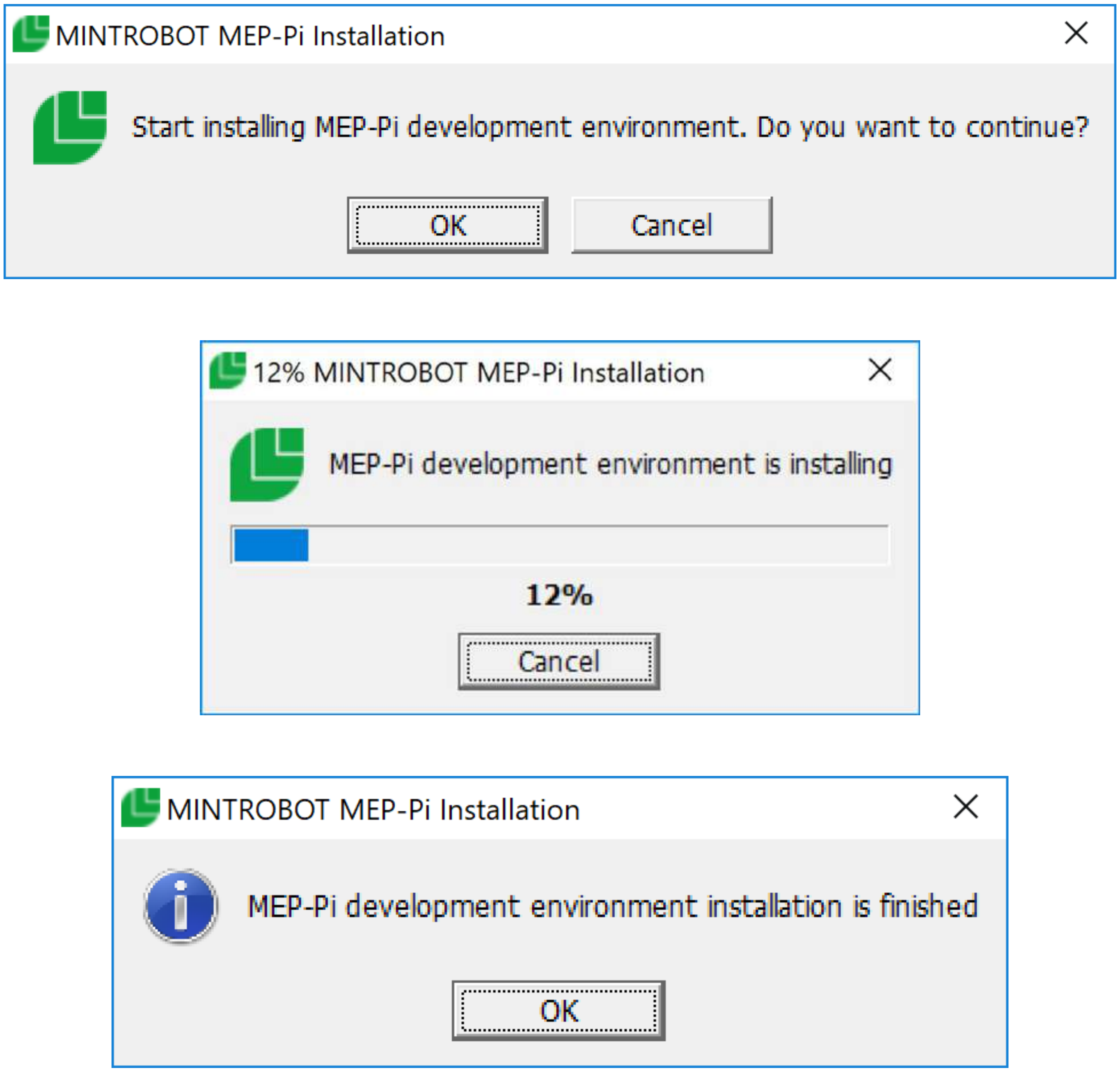
<Figure 18> Connection to SAMBA server

Now, users can transmit files to MEP-Pi as same way they did in Windows system. Creation of new files and folder are also same. It is useful when transmit built binary files from IDE. We will discuss about this at next chapter.

3. Development environment setup

3.1. Install software

First, execute installer file in the provided USB. Then, the installation of MEP-Pi development environment is started. **NOTE, the installation path is "C:\MINTROBOT\MEP-Pi\", and it is fixed. Please, check there is sufficient space left for the installation. The installation requires at least 4GB space.** If you click OK, installer starts installation.



MINTROBOT MEP-Pi Installation

Start installing MEP-Pi development environment. Do you want to continue?

12% MINTROBOT MEP-Pi Installation

MEP-Pi development environment is installing

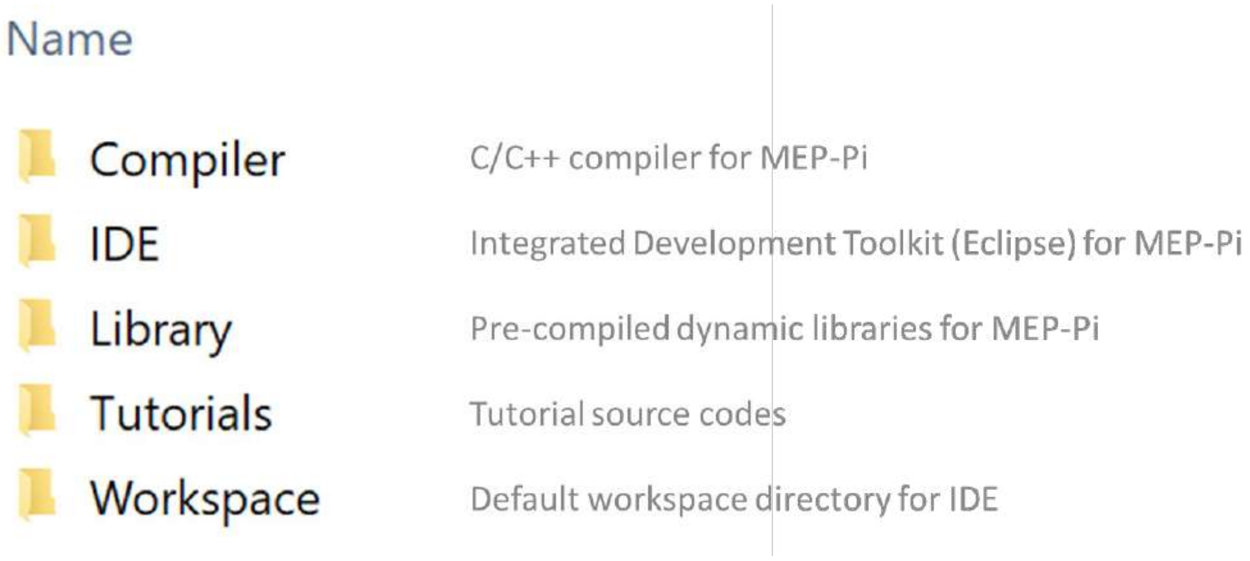
12%

MINTROBOT MEP-Pi Installation

MEP-Pi development environment installation is finished

<Figure 19> Installation screen

3.2 Software details



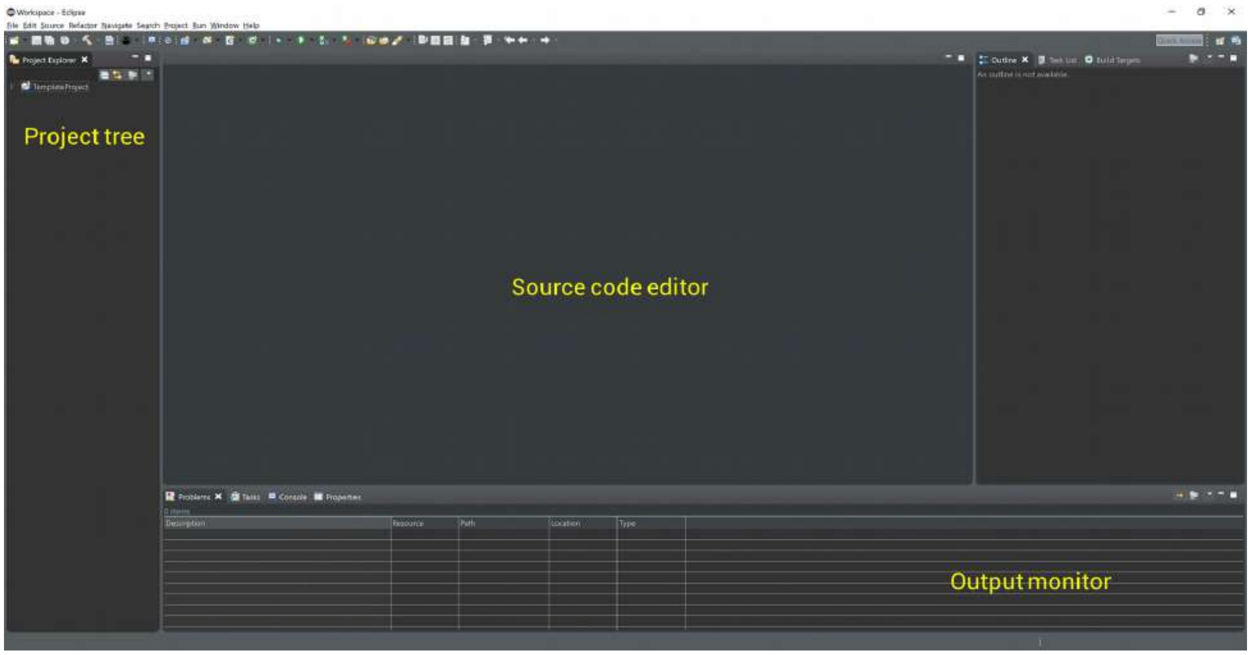
Name	
Compiler	C/C++ compiler for MEP-Pi
IDE	Integrated Development Toolkit (Eclipse) for MEP-Pi
Library	Pre-compiled dynamic libraries for MEP-Pi
Tutorials	Tutorial source codes
Workspace	Default workspace directory for IDE

<Figure 20> Folders of the installed software

Compiler	arm-linux-gnueabihf compiler
IDE	Eclipse IDE for C/C++ developers (CDT)
Library	Pre-compiled libraries and header files including 3rd parties' (Etherlab IgH, OROCOS KDL, eigen)
Tutorials	Step by step example codes
Workspace	Default location to store source code used in IDE

3.3 Getting started with IDE

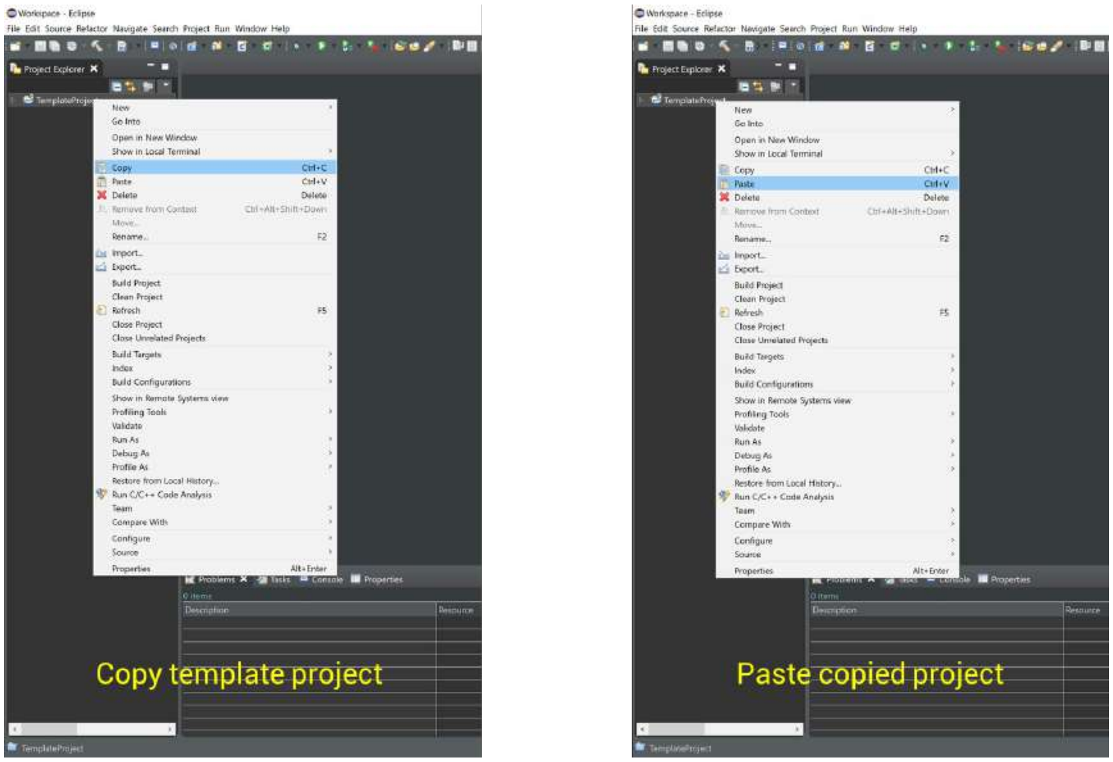
Enter the IDE folder and execute eclipse.exe, then eclipse IDE for MEP-Pi is executed. Provided Eclipse was customized for the development of application for MEP-Pi. User can see only "TemplateProject" at the project tree view at the first execution. It is the basic template project for MEP-Pi application. Every environmental settings is preceded for convenience. **NOTE, do not use the template project and do not modify the template project. Do clone (copy & paste) the template project to make new project.**



<Figure 21> First screen of IDE

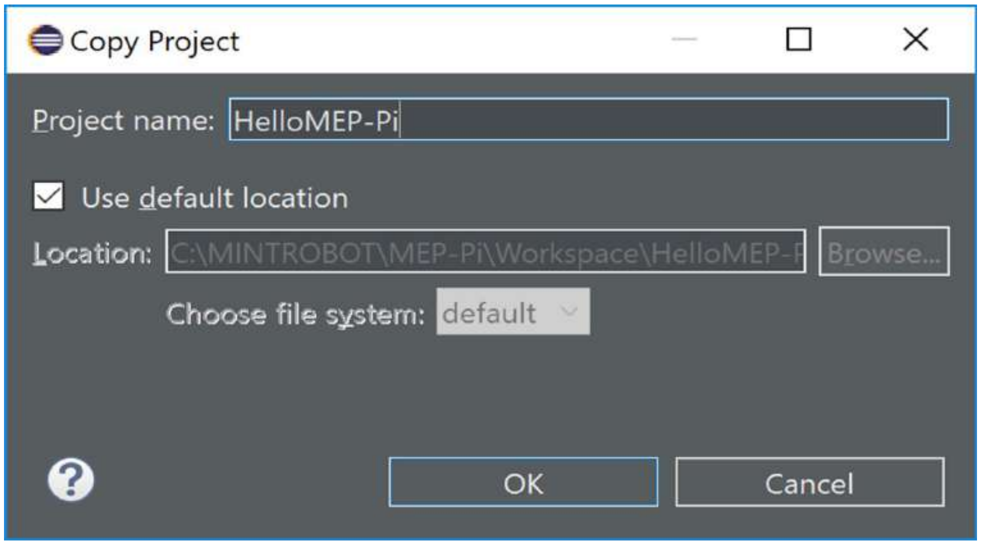
3.4 Hello world example

3.4.1 Make a clone project



<Figure 22> Processes to clone the template project

First, make the clone project by using the template project. Copy the template project, and past it at the project tree. Then, name the new project as “Hello MEP-Pi”. Then, new project is generated in the project tree with included basic header files.


Copy Project

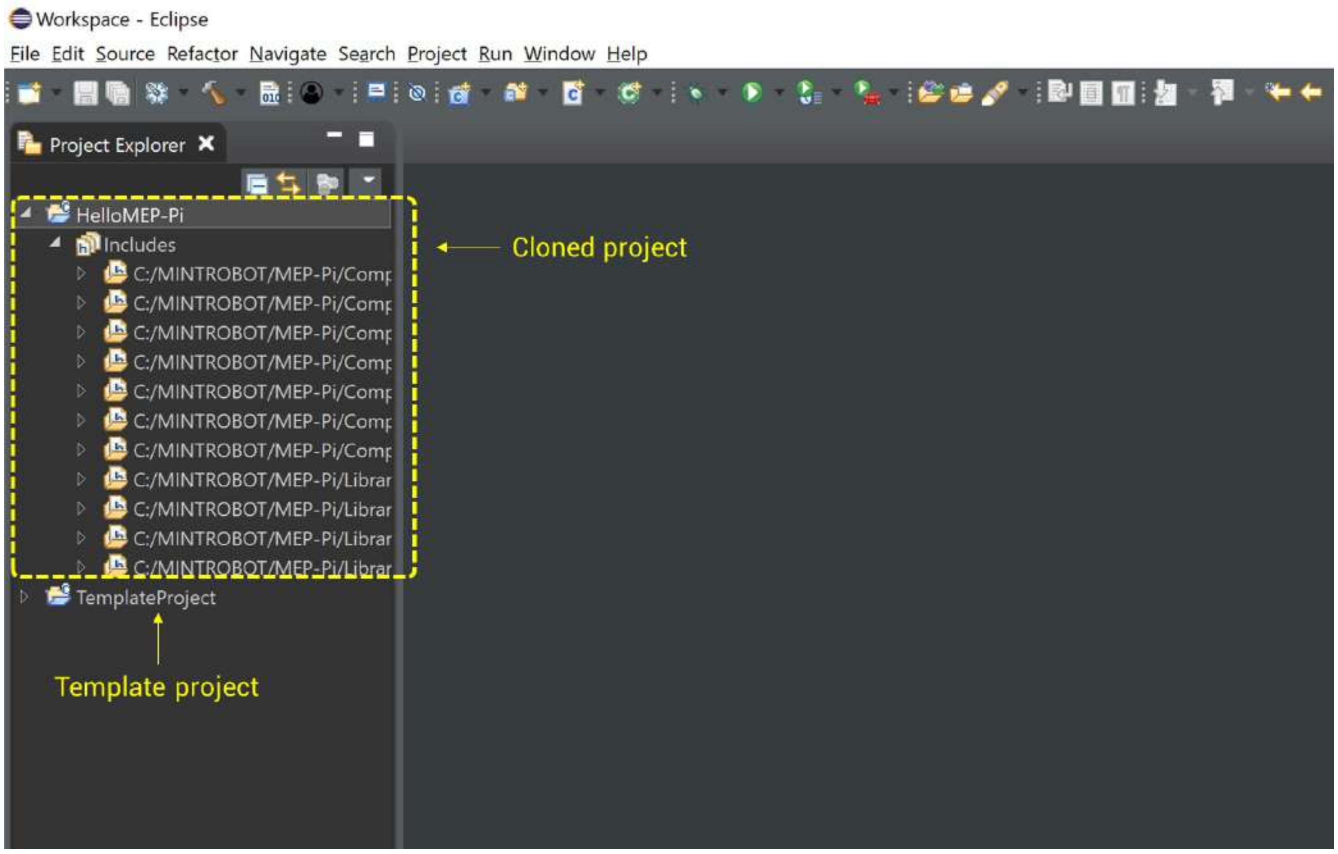
Project name: HelloMEP-Pi

☑ Use default location

Location: C:\MINTROBOT\MEP-Pi\Workspace\HelloMEP-P

Choose file system: default

<Figure 23> Naming of the clone project


Workspace - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer

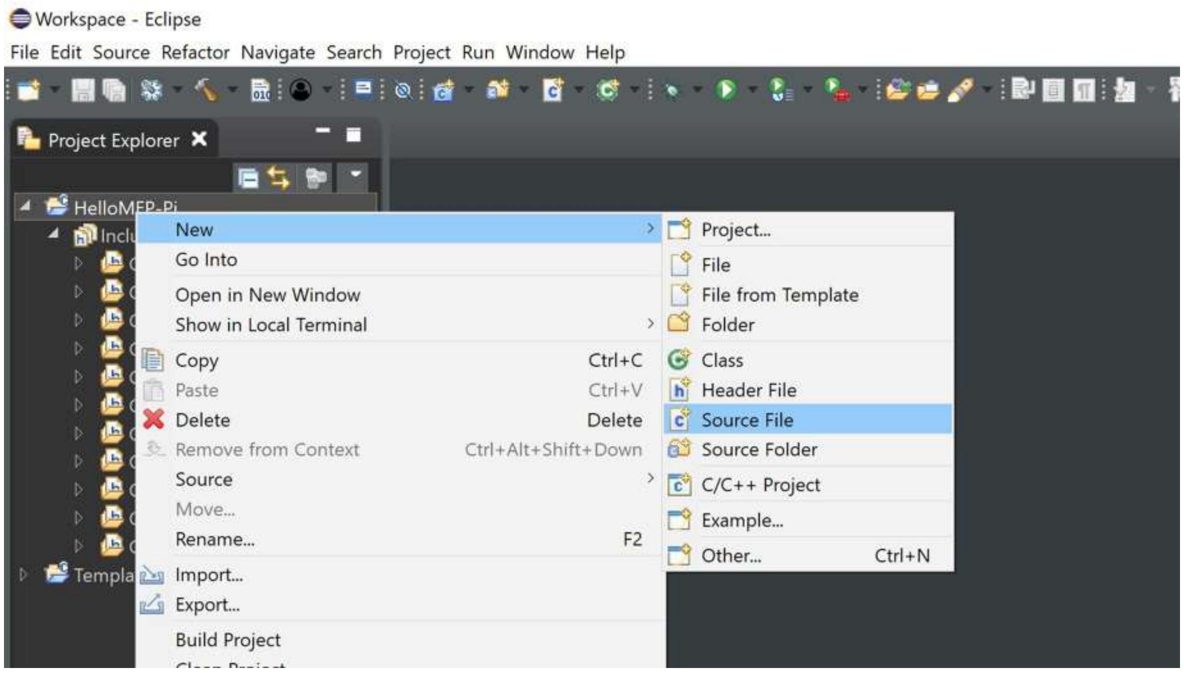
HelloMEP-Pi
Includes
C:/MINTROBOT/MEP-Pi/Comp
C:/MINTROBOT/MEP-Pi/Comp
C:/MINTROBOT/MEP-Pi/Comp
C:/MINTROBOT/MEP-Pi/Comp
C:/MINTROBOT/MEP-Pi/Comp
C:/MINTROBOT/MEP-Pi/Comp
C:/MINTROBOT/MEP-Pi/Comp
C:/MINTROBOT/MEP-Pi/Librar
C:/MINTROBOT/MEP-Pi/Librar
C:/MINTROBOT/MEP-Pi/Librar
C:/MINTROBOT/MEP-Pi/Librar
TemplateProject

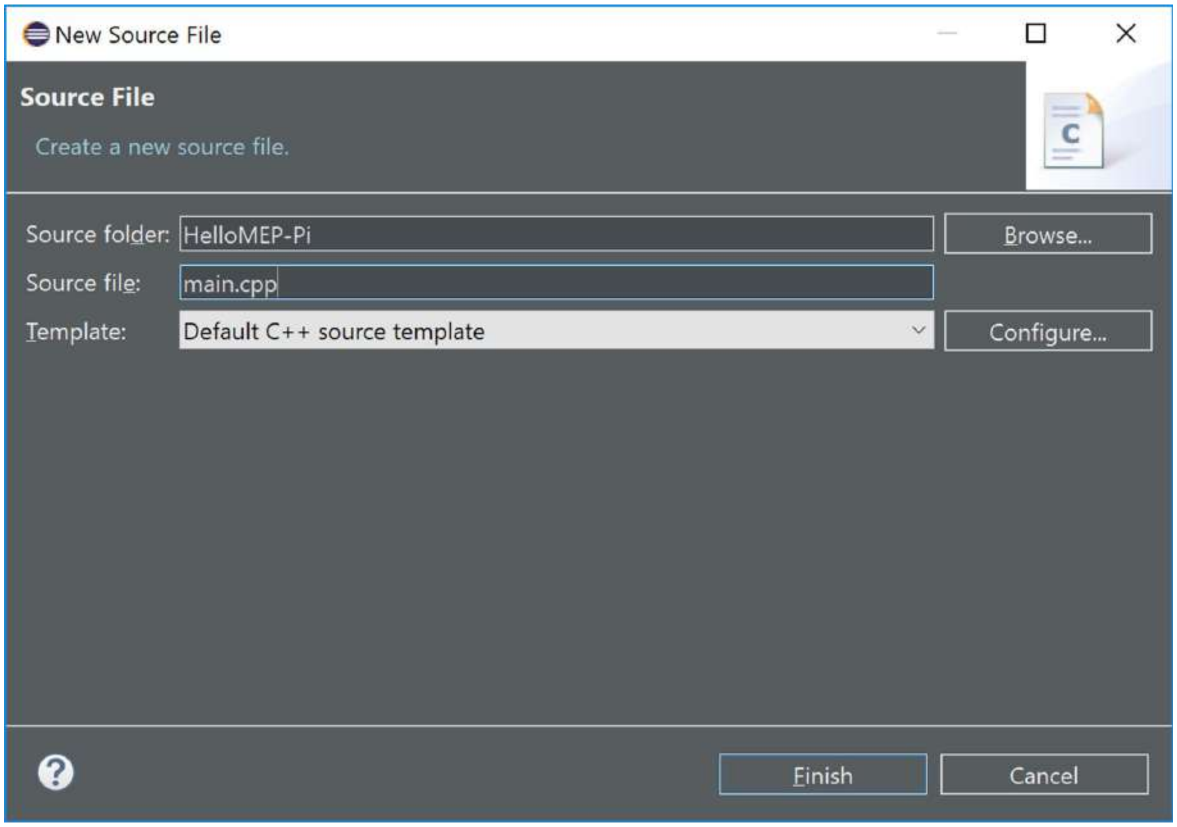
Cloned project
Template project

<Figure 24> Generated clone project

3.4.2 Add new source file

Right click at the new project, and click “New → Source File”. Then, name the new file to main.cpp.

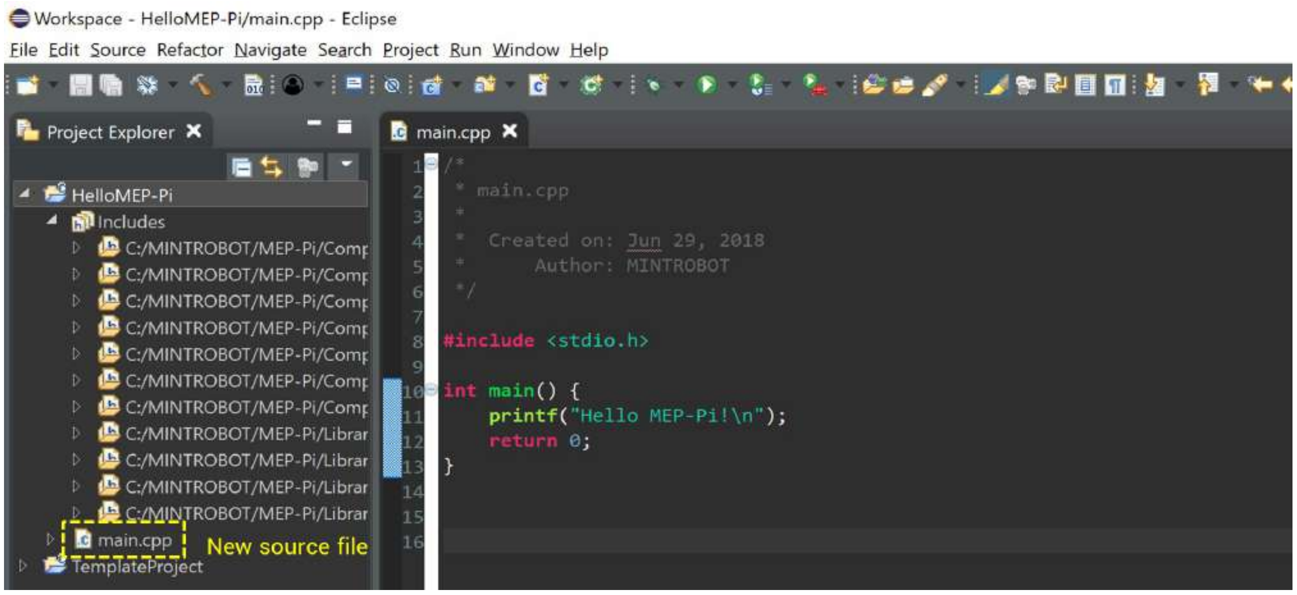




Source folder: HelloMEP-Pi
Source file: main.cpp
Template: Default C++ source template

<Figure 25> Add new “main.cpp” source file

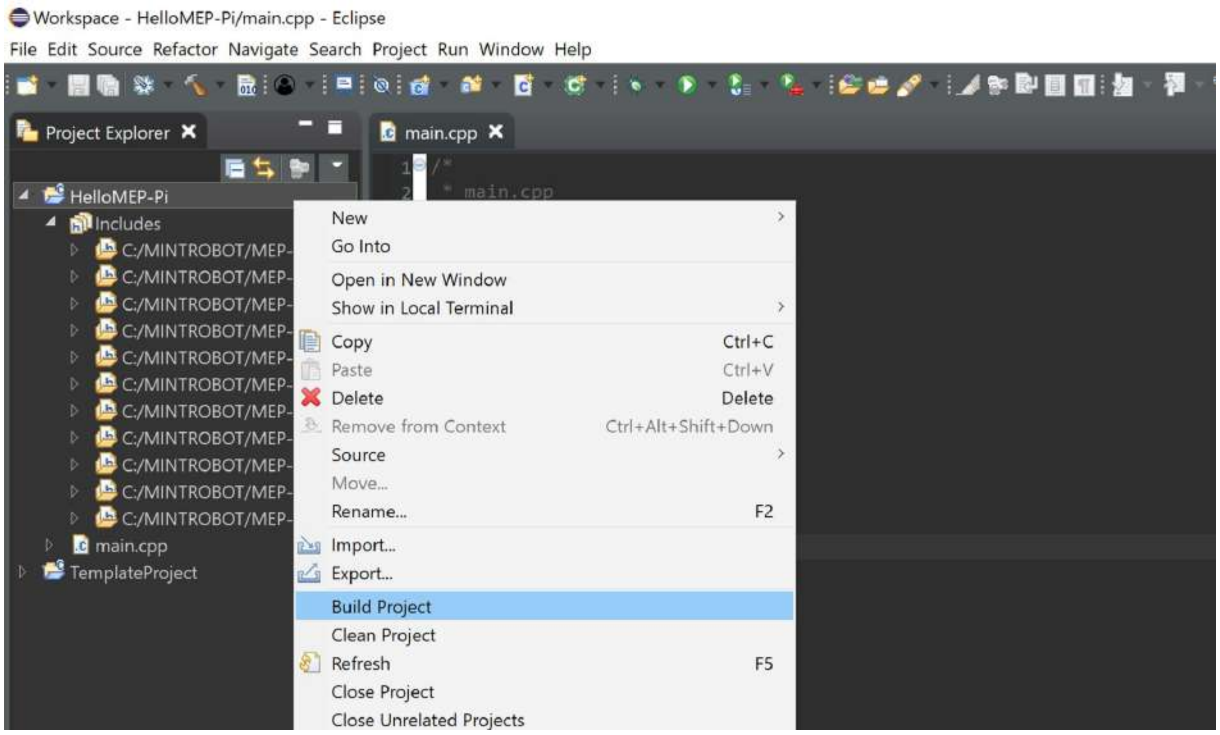
The new source file "main.cpp" is added in the project tree. Then, user can edit the code of the "main.cpp" by using the source code editor. Type the sample source code as bellows.



```
/*
 * main.cpp
 *
 *  Created on: Jun 29, 2018
 *      Author: MINTROBOT
 */

#include <stdio.h>

int main() {
    printf("Hello MEP-Pi!\n");
    return 0;
}
```

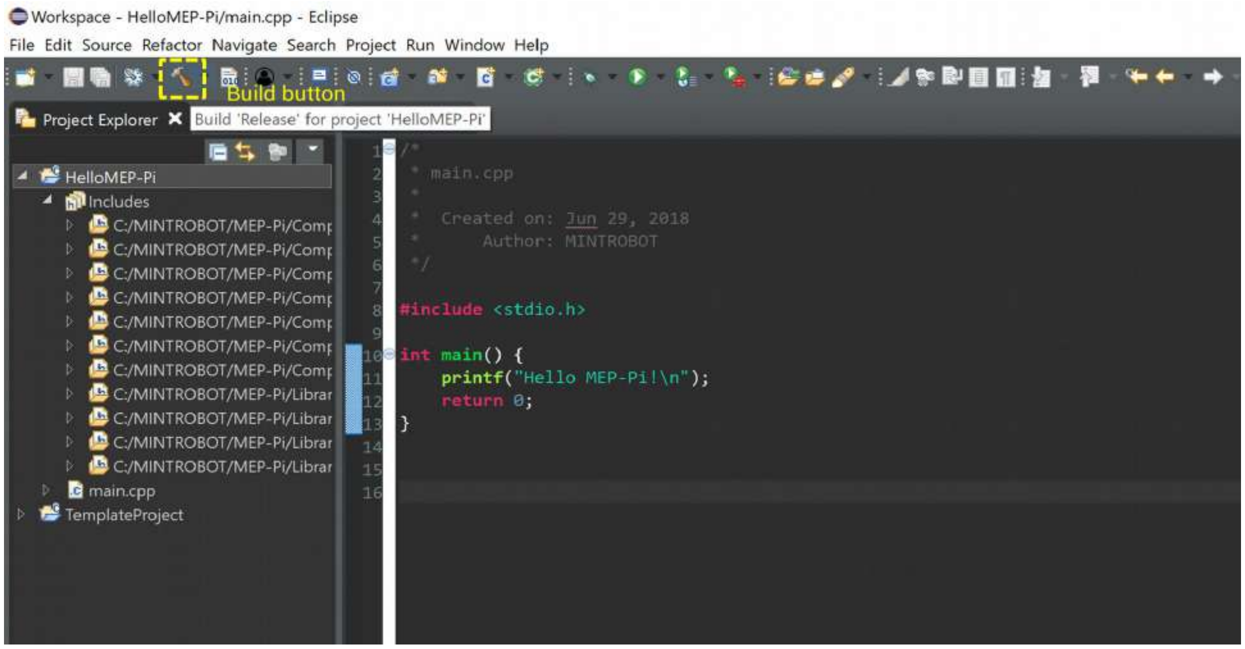
<Figure 26> "main.cpp" source file contents

3.4.3 Build project



<Figure 27> Build project from popup menu

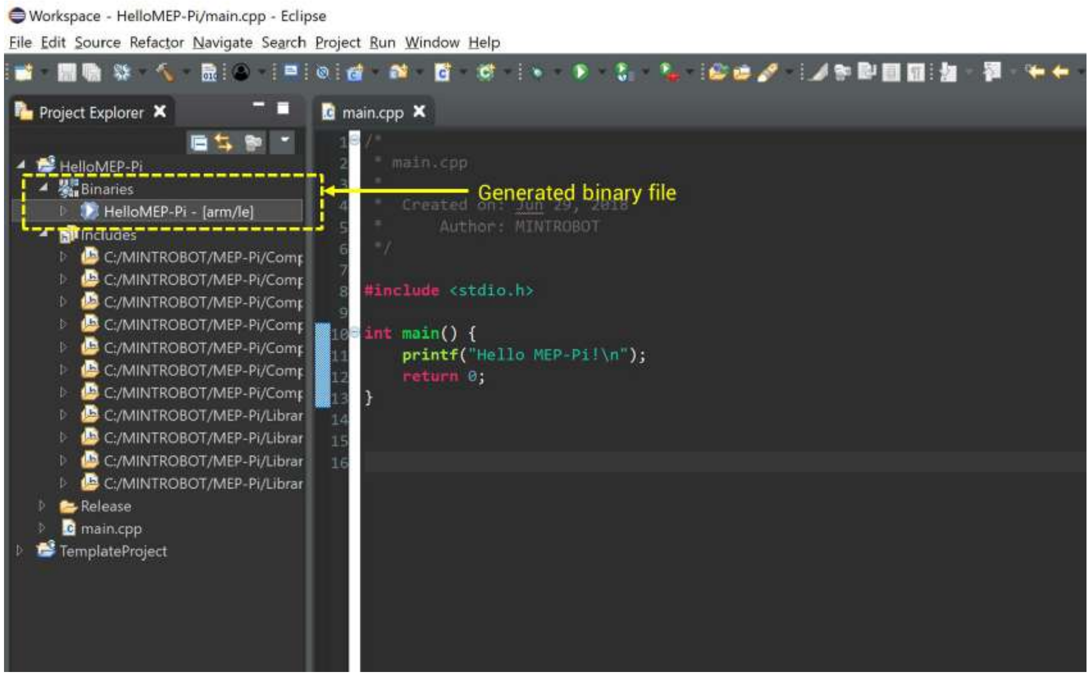
There are two way to build the project. 1). Right click on the project tree and select "Build Project" 2). select the target project on the project and click "Build button" on the toolbar.


Workspace - HelloMEP-Pi/main.cpp - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Build button
Project Explorer
Build 'Release' for project 'HelloMEP-Pi'
HelloMEP-Pi
Includes
C:/MINTROBOT/MEP-Pi/Comp
C:/MINTROBOT/MEP-Pi/Librar
main.cpp
TemplateProject

```
/*
 * main.cpp
 *
 *  Created on: Jun 29, 2018
 *      Author: MINTROBOT
 */

#include <stdio.h>

int main() {
    printf("Hello MEP-Pi!\n");
    return 0;
}
```

<Figure 28> Build project from toolbar icon

If there are no build errors, the final binary file is generate under "Binaries" slot on the project tree. It is an executable file under the operating system of MEP-Pi.



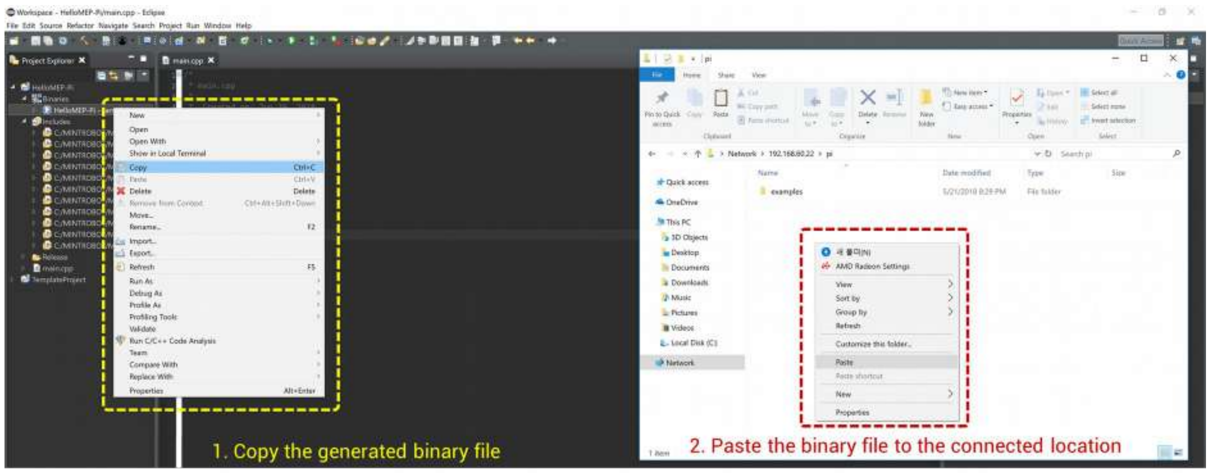
```
/*
 * main.cpp
 *
 *  Created on: Jun 29, 2018
 *      Author: MINTROBOT
 */

#include <stdio.h>

int main() {
    printf("Hello MEP-Pi!\n");
    return 0;
}
```

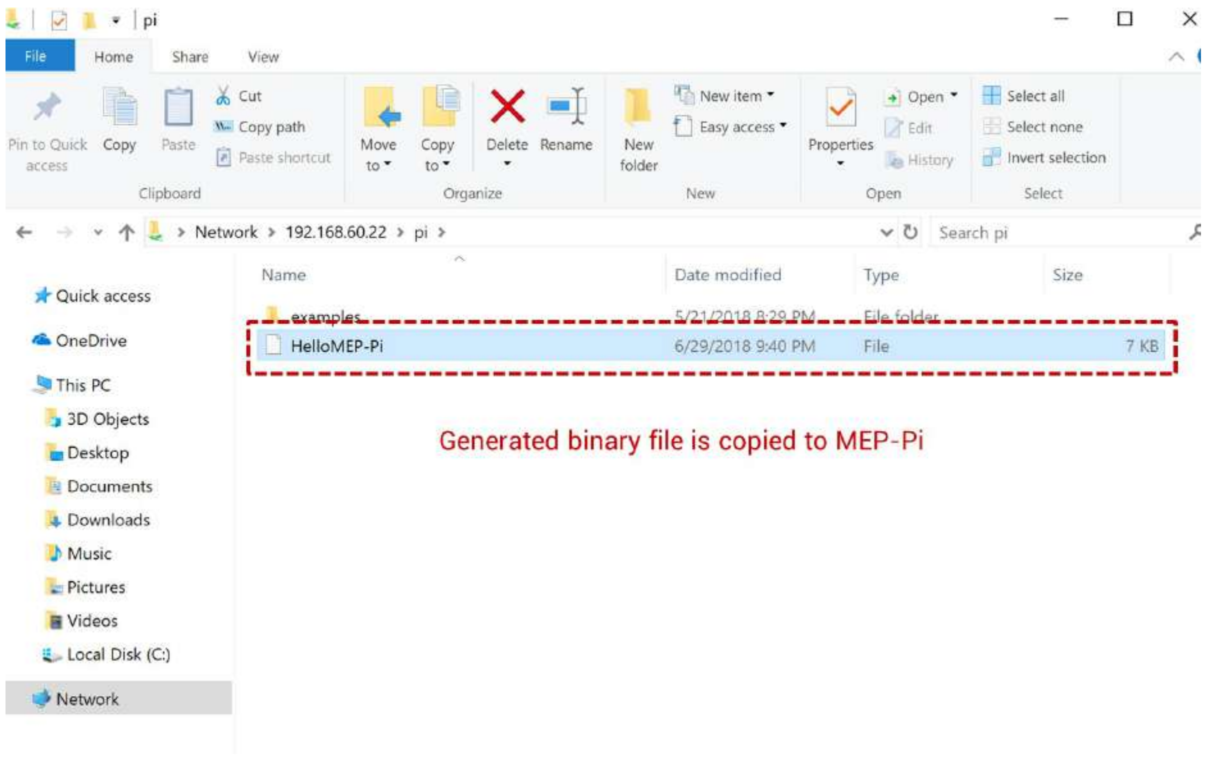
<Figure 29> Finally built binary file from building process

3.4.4 Run with MEP-Pi

There are many method to run built binary file linux system as like MEP-Pi, but we officially recommend the method as bellows.

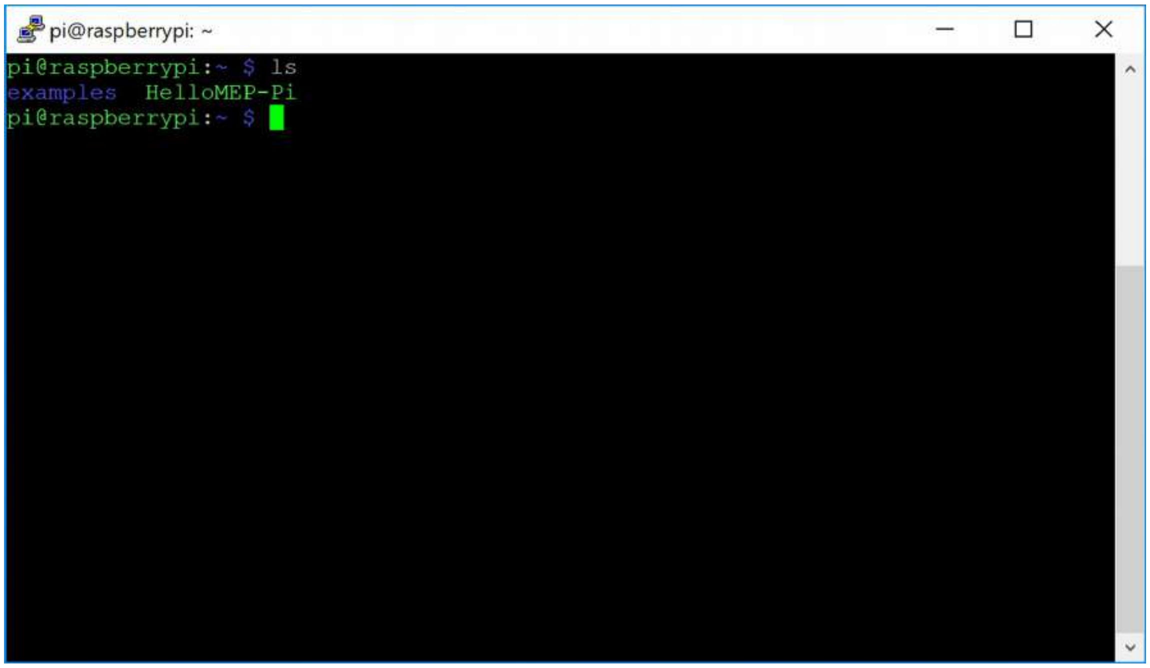
1) Connect MEP-Pi by using samba server by using "ID : pi / PASS : raspberry". The default location is "/home/pi". and copy the generated binary file to the connected location.


1. Copy the generated binary file
2. Paste the binary file to the connected location


Network > 192.168.60.22 > pi
Name | Date modified | Type | Size
examples | 5/21/2018 8:29 PM | File folder
HelloMEP-Pi | 6/29/2018 9:40 PM | File | 7 KB
Generated binary file is copied to MEP-Pi

<Figure 30> Copy binary file to MEP-i by using SAMBA server

2) Connect MEP-Pi by using putty by using “ID : pi / PASS : raspberry”. Then, type “ls” to verify copied file is correctly located.

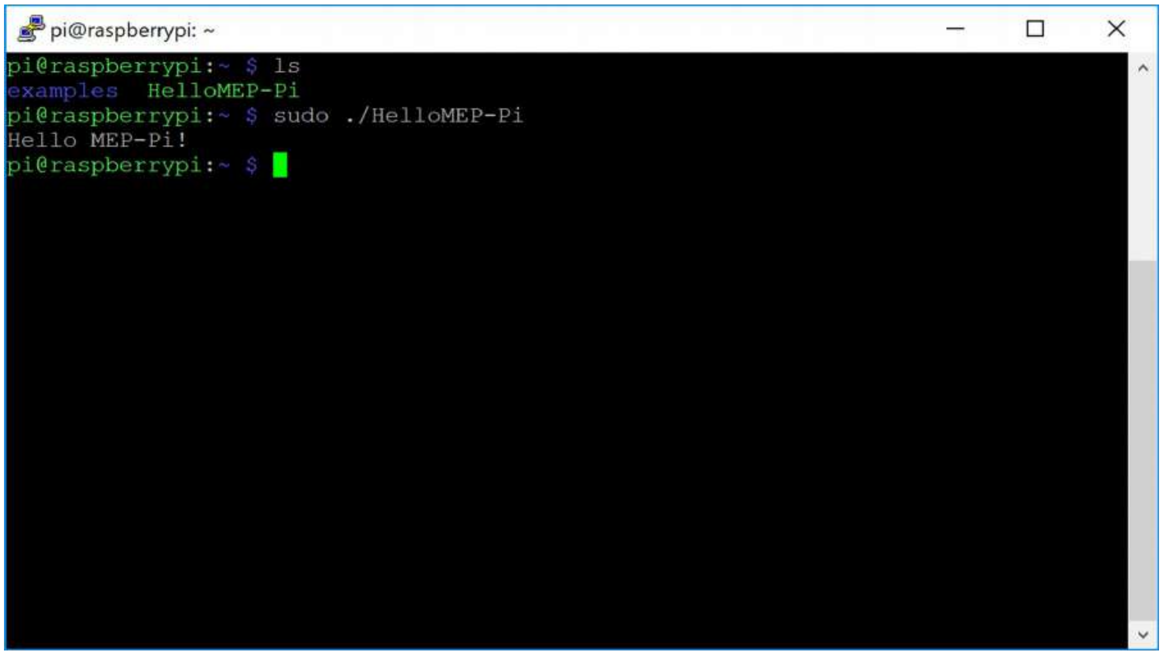


pi@raspberrypi: ~

```
pi@raspberrypi:~ $ ls
examples  HelloMEP-Pi
pi@raspberrypi:~ $
```

<Figure 31> Remote connection and verification of the copied binary file

3) Run the binary file with “sudo”. The application of MEP-Pi requires root authority because it uses real-time functions. Every application from the template project must run with "sudo" command.



pi@raspberrypi: ~

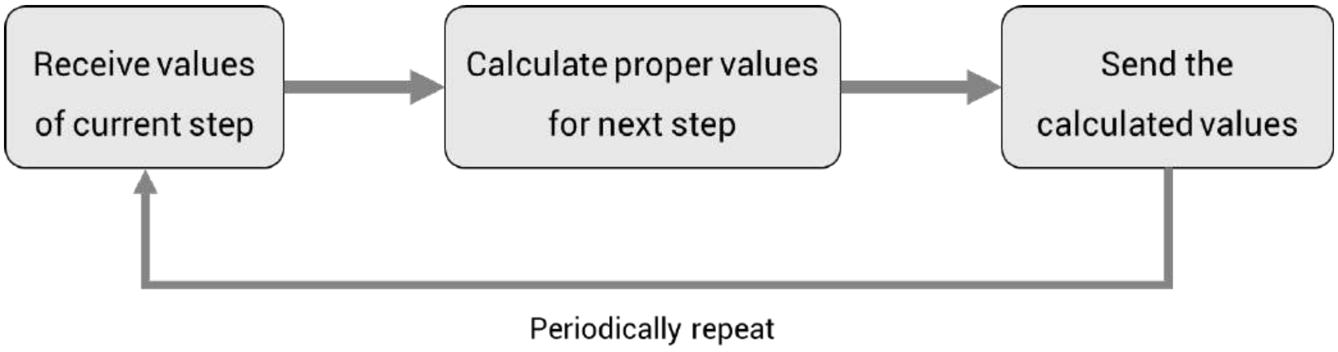
```
pi@raspberrypi:~ $ ls
examples  HelloMEP-Pi
pi@raspberrypi:~ $ sudo ./HelloMEP-Pi
Hello MEP-Pi!
pi@raspberrypi:~ $
```

<Figure 32> Execution of the copied binary file

4. Tutorials

4.1 Basic concept of real-time based motion control

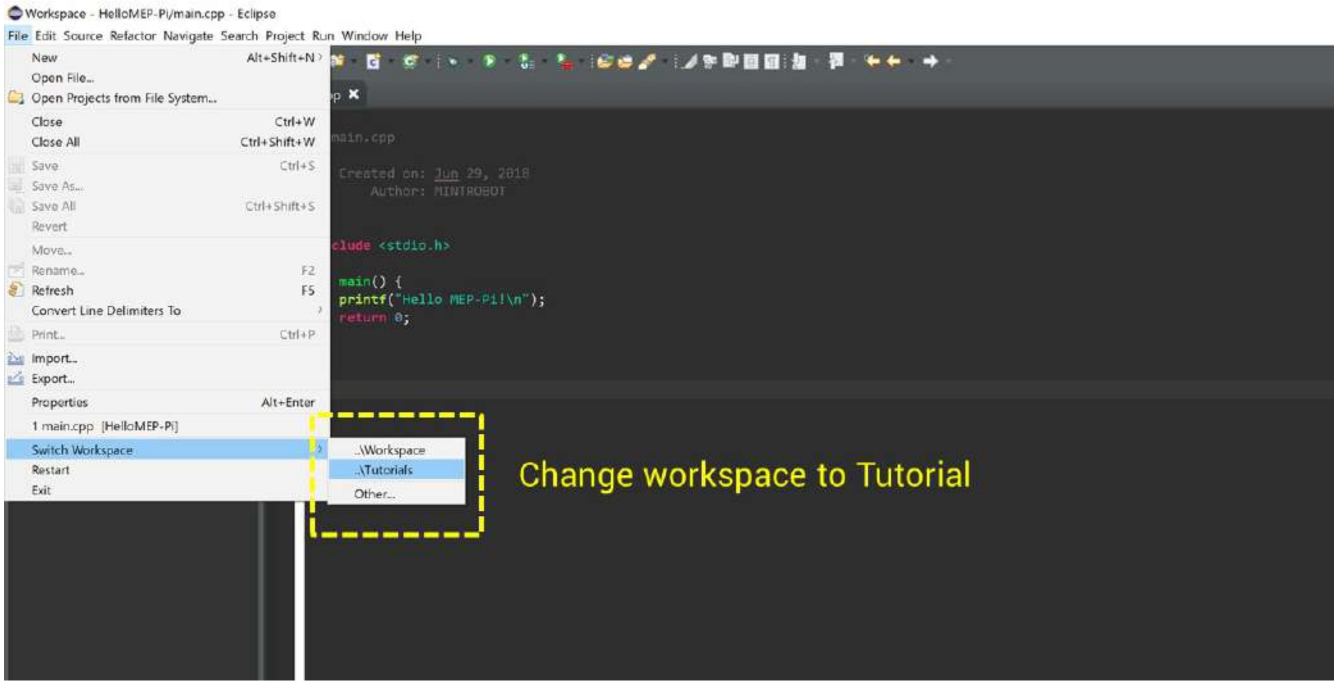
MEP-Pi is a real-time capable motion control platform. The fundamental concept of the real-time based motion control is that it calculates proper values for the motion control in every cycle. And, the real-time means that the periodic calculation is guaranteed. **NOTE, the real-time capable does not mean that it can guarantee it can finish the processing of the given work regardless of the computing power. It does mean that it can guarantee it can start the given work in every periodic.** Therefore, the proper selection of the period of the given task with the consideration of the computing power is very important.


Receive values of current step
Calculate proper values for next step
Send the calculated values
Periodically repeat

<Figure 33> Basic process of real-time based motion control

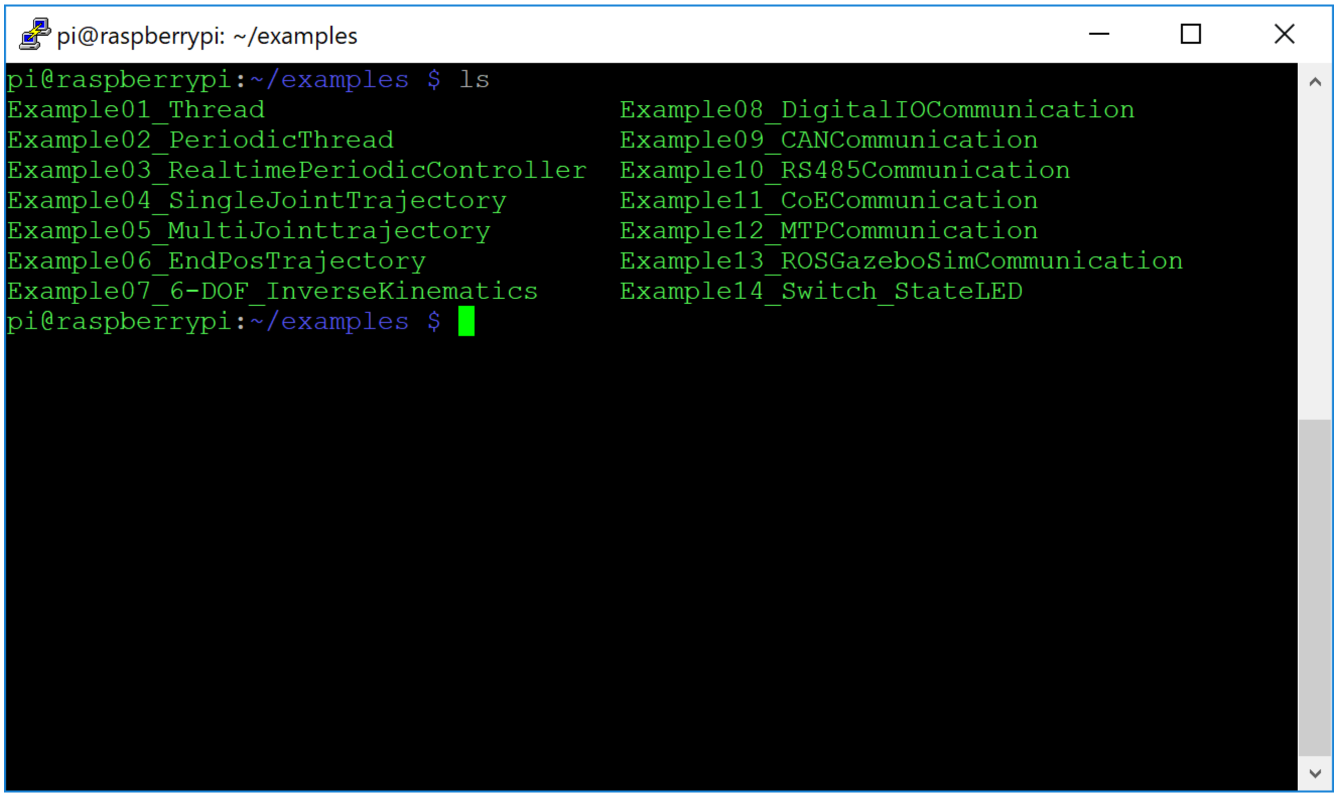
4.2 Step by step examples

The provided step by step examples represent the process to implement motion controller for a 6-DOF robot arm. First, it shows the example codes to understand the structure of a real-time based motion controller. Next, it shows the example codes to understand how to calculate the proper values of motion control in terms of position control, such as trajectory and kinematics. Then, it shows the example codes for Digital In/Out, CAN, RS485, and EtherCAT. Lastly, user can test the implemented motion control application with simulator and MTP that is a freely provided teaching pendant software. The example codes are stored in Tutorial folder of the installed software. Users can easily load the example codes by switching workspace of IDE as shown in fig. 34. There are many comments in the sources code, so user can easily understand the structure of each examples. Also, all example codes are pre-compiled and stored in MEP-Pi (/home/pi/examples).


Workspace - HelloMEP-Pi/main.cpp - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Switch Workspace
..\Workspace
..\Tutorials
Other...
Change workspace to Tutorial

```
clude <stdio.h>

main() {
printf("Hello MEP-Pi!\n");
return 0;
```

<Figure 34> Switching workspace for tutorial



```
pi@raspberrypi:~/examples $ ls
Example01_Thread                        Example08_DigitalIOCommunication
Example02_PeriodicThread                Example09_CANCommunication
Example03_RealtimePeriodicController    Example10_RS485Communication
Example04_SingleJointTrajectory         Example11_CoECommunication
Example05_MultiJointtrajectory          Example12_MTPCommunication
Example06_EndPosTrajectory              Example13_ROSGazeboSimCommunication
Example07_6-DOF_InverseKinematics       Example14_Switch_StateLED
pi@raspberrypi:~/examples $
```

<Figure 35> Pre-compiled example codes

1.3.1 Thread

In this example, user can study how to create threads with priority. Users can create two threads that compete concurrently to gain computing power according to their priority.

1.3.2 Periodic thread

In this example, user can study how to create thread that repeat its functions periodically. Users can study how to set set the period for the periodic class, and can verify it repeats its functions with given period.

1.3.3 Real-time periodic controller

In this example, user can study the implementation details of a real-time based motion control software as shown in fig. 33. User can verify how the periodic control algorithm is implemented.

1.3.4 Single joint trajectory

In this example, user can study how to calculate proper angle for a single joint. The examples represents the method to generate trajectory according to maximum velocity and acceleration. And, it shows how to get the proper value of the joint from the calculated trajectory by using time variable.

1.3.5 Multi joint trajectory

In this example, user can study how to calculate proper angle for multi joints. It is extended example from the single joint trajectory example in order to implement multi-joint robot arm. Users can study the implementation details for joint angle based motion controller after finishing this example. This example can be used as the motion controller for XYZ 3-axis linear motion system, such as CNC milling and 3D printer.

1.3.6 End-pos trajectory

In this example, users can study how to calculate proper trajectory of an end-point, which is the trajectory of the end point of a robot arm.

1.3.7 6-DOF inverse kinematics

In this example, user can study how to calculate proper angle for a 6-DOF robot arm according to the calculated end-pos trajectory at example 1.3.6. It means user can convert the XYZ space based value into joint space based value, which will be sent to the motor of the joint. User can study the implementation details for a motion controller for a robot arm, such as 6-DOF robot manipulators.

1.3.8 Digital IO communication

In this example, user can study how to use the APIs in order to send and receive values from digital in/out interface. It can be used when users need to integrate their motion control application with PCL-like higher system.

1.3.9 CAN communication

In this example, user can study how to use the APIs in order to communicate with CAN devices. The source codes implemented the communication with 6-DOF F/T sensor of ROBOTOUS (www.robotous.com) which can be used for direct teaching algorithm.

1.3.10 RS485 communication

In this example, user can study how to use the APIs in order to communicate with RS485 devices. The source codes implemented the communication example with Holloint Joint, which is the smart actuator of MINTROBOT.

1.3.11 EtherCAT CoE communication

In this example, user can study how to use the APIs in order to implement EtherCAT Master. The source codes implemented the control of 6 x Panasonic servo motors, which assume the motor for 6-DOF robot arm.

1.3.12 Communication with MTP

In this example, user can study how to integrate the motion controller with PC based higher system, such as teaching pendant. MTP is a freely provided teaching pendant software that can program robot arms easily. The communication between MEP-Pi and MTP is TCP/IP communication, and user can integrate their application with other software by using same manner.

1.3.13 Communication with ROS gazebo simulator

In this example, user can integrate the implemented motion control application with a simulator. User can verify how the robot moves in simulation environment.

1.3.14 Switch state LEDs

In this example, user can study how to handle the LEDs to indicate state of the motion controller. User can inform such as warning and errors by using LEDs.