**MODUL PEMBUATAN PEMBUATAN MINI SCADA HMI DENGAN BAHASA PEMROGRAMAN PYTHON DAN QML UNTUK PLC OUTSEAL**

****

****

Disusun oleh:

Muhammad Husni Muttaqin, S. Pd.

**Lab. Mekatronika Politeknik Elektronika Negeri Surabaya**

**dan Komunitas Belajar Arduino dan Mekatronika**

**2023**

# Kata Pengantar

Segala puji bagi Allah, Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan modul ajar. Tak lupa juga mengucapkan salawat serta salam semoga senantiasa tercurahkan kepada Nabi Besar Muhammad SAW, karena berkat beliau, kita mampu keluar dari kegelapan menuju jalan yang lebih terang.

Adapun, modul kami yang berjudul ‘**PEMBUATAN MINI SCADA HMI DENGAN BAHASA PEMROGRAMAN PYTHON DAN QML UNTUK PLC OUTSEAL’** ini telah selesai kami buat secara semaksimal dan sebaik mungkin agar menjadi manfaat bagi pembaca yang membutuhkan informasi dan pengetahuan mengenai bagaimana pembuatan MINI HMI SCADA untuk PLC outseal.

Dalam modul ini, tertulis bagaimana langkah-langkah pembuatan MINI HMI SCADA untuk PLC outseal dari pembuatan base code hingga bagaimana cara mengintegrasikan ke PLC outseal via MODBUS RTU yang disajikan secara jelas dan detail.

Kami sadar, masih banyak luput dan kekeliruan yang tentu saja jauh dari sempurna tentang modul ini. Oleh sebab itu, kami mohon agar pembaca memberi kritik dan juga saran terhadap karya modul ini agar kami dapat terus meningkatkan kualitas modul.

Demikian modul ini kami buat, dengan harapan agar pembaca dapat memahami informasi dan juga mendapatkan wawasan mengenai pembuatan MINI HMI SCADA untuk PLC outseal serta dapat bermanfaat bagi masyarakat dalam arti luas dan dapat menjadi referensi untuk inovasi-inovasi di bidang otomasi dan pemrograman. Terima kasih.

Daftar Isi

[Kata Pengantar 1](#_Toc124263755)

[Daftar Isi 3](#_Toc124263756)

[Persiapan 5](#_Toc124263757)

[Pengenalan Software Tools 5](#_Toc124263758)

[Notepad++ 5](#_Toc124263759)

[Thonny IDE 6](#_Toc124263760)

[Outseal Studio 8](#_Toc124263761)

[Instalasi Library Python 8](#_Toc124263762)

[Install PyQt5 8](#_Toc124263763)

[Install Pymodbus 9](#_Toc124263764)

[Wiring Hardware 12](#_Toc124263765)

[Jobsheet 0 : Base Code 13](#_Toc124263766)

[Program Python 13](#_Toc124263767)

[Program QML 16](#_Toc124263768)

[Jobsheet 1 : Digital Output 18](#_Toc124263769)

[Single Output 18](#_Toc124263770)

[Program Ladder 18](#_Toc124263771)

[Program Python 18](#_Toc124263772)

[Program QML 22](#_Toc124263773)

[Multiple Output 25](#_Toc124263774)

[Program Ladder 25](#_Toc124263775)

[Program Python 25](#_Toc124263776)

[Program QML 29](#_Toc124263777)

[Jobsheet 2 : Digital Input 34](#_Toc124263778)

[Single Input 34](#_Toc124263779)

[Program Ladder 34](#_Toc124263780)

[Program Python 34](#_Toc124263781)

[Program QML 38](#_Toc124263782)

[Multiple Input 42](#_Toc124263783)

[Program Ladder 42](#_Toc124263784)

[Program Python 42](#_Toc124263785)

[Program QML 47](#_Toc124263786)

[Jobsheet 3 : Analog Input 51](#_Toc124263787)

[Basic 51](#_Toc124263788)

[Program Ladder 51](#_Toc124263789)

[Program Python 51](#_Toc124263790)

[Program QML 55](#_Toc124263791)

[Customize 58](#_Toc124263792)

[Program Ladder 58](#_Toc124263793)

[Program Python 58](#_Toc124263794)

[Program QML 62](#_Toc124263795)

[Jobsheet 4 : Analog Output 67](#_Toc124263796)

[Basic 67](#_Toc124263797)

[Program Ladder 67](#_Toc124263798)

[Program Python 67](#_Toc124263799)

[Program QML 71](#_Toc124263800)

[Customize 73](#_Toc124263801)

[Program Ladder 73](#_Toc124263802)

[Program Python 73](#_Toc124263803)

[Program QML 77](#_Toc124263804)

[Jobsheet 5 : Historical Graph 82](#_Toc124263805)

[Program Ladder 82](#_Toc124263806)

[Program Python 82](#_Toc124263807)

[Program QML 86](#_Toc124263808)

# 

# Persiapan

## Pengenalan Software Tools

### Notepad++

Notepad++ adalah source code editor gratis sebagai pengganti notepad yang mendukung beberapa bahasa pemrograman. Notepad++ ditulis dalam Bahasa C++ dan menggunakan Win32 API murni dan STL yang memastikan kecepatan eksekusi yang lebih tinggi, ukuran program yang lebih kecil dan penggunaan CPU yang lebih ringan [17]. Cara instal Notepad++ adalah dengan mengakses link <https://notepad-plus-plus.org/downloads/> lalu download installer yang paling atas (artinya paling baru) dan lakukan penginstalan.

Graphical user interface, text, application

Description automatically generated

Berdasarkan gambar diatas, klik yang paling atas lalu download dan instal. Setelah instal, tampilannya sebagai berikut:

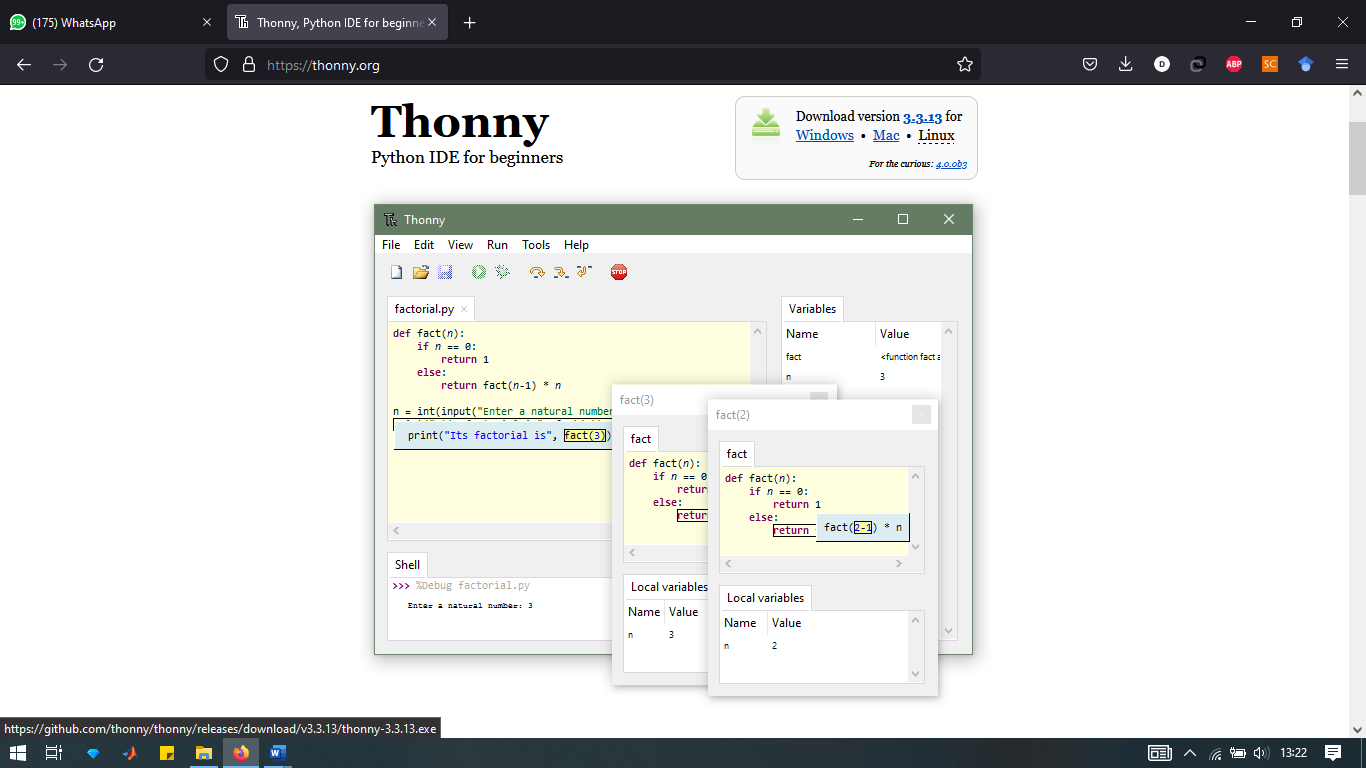
Graphical user interface, application, Word

Description automatically generated

Notepad++ nantinya akan digunakan untuk menulis code dalam ekstensi qml.

### Thonny IDE

Thonny adalah software yang dikembangkan oleh University of Tartu di Estonia yang diperuntukkan untuk programmer python pemula. Thonny merupakan software open source alias gratis dan memiliki lisensi GPL v3 [16]. Thonny sudah memiliki Python 3.7 didalamnya yang akan memudahkan pengguna untuk mengeksekusi program dengan cepat. Untuk mengintstal Thonny IDE, kita dapat mengakses link <https://thonny.org/>, lalu memilih system operasi sesuai dengan yang kita gunakan. Ada 3 opsi system operasi yaitu Windows, MacOS, dan Linux. Tampilan laman thonny.org sebagai berikut:



Berdasarkan gambar diatas, klik download version sesuai system operasi kita dan lakukan penginstalan sampai selesai. Setelah selesai proses instal, tampilan Thonny IDE seperti berikut:

Graphical user interface, application, Teams

Description automatically generated

Gambar 6. Tampilan awal Thonny IDE.

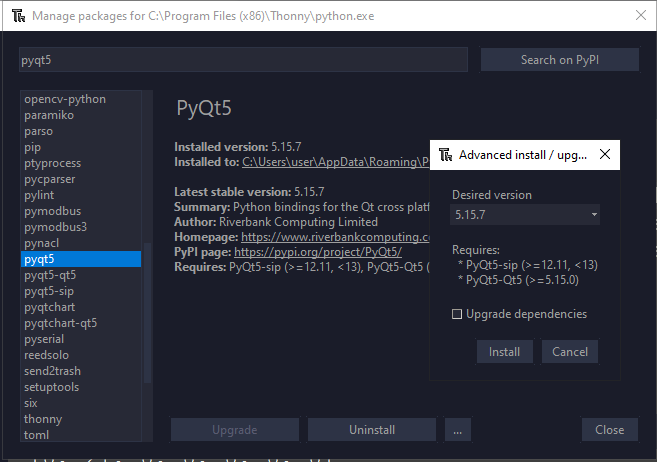
Berdasarkan gambar diatas, kolom atas adalah untuk kita menulis coding dan kolom bawah (shell) adalah hasil running coding akan muncul disitu.

### Outseal Studio

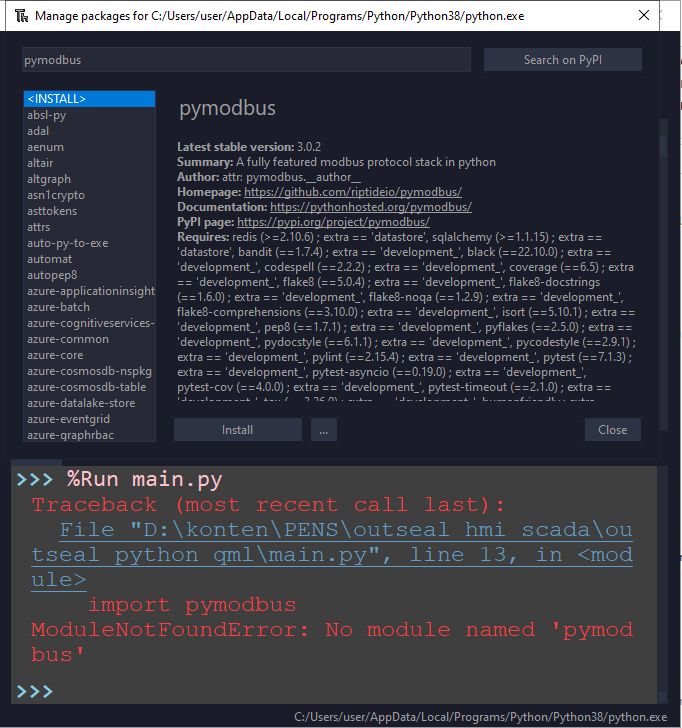
Outseal adalah sebuah teknologi otomasi karya anak bangsa. Produk outseal berupa PLC (Progammable Logic Controller), HMI(Human Machine Interface) dan modul-modul yang lain. PLC outseal dibuat berbasis arduino bootloader dan desain hardware nya dibuka untuk umum, artinya anda bisa download dan mempelajari rangkaian elektroniknya secara bebas serta membuat sendiri di rumah menggunakan papan mikrokontroller arduino dengan harga yang terjangkau. Dan yang menarik adalah software nya berupa program visual (ladder diagram), berbahasa Indonesia dan juga dibuat gratis.

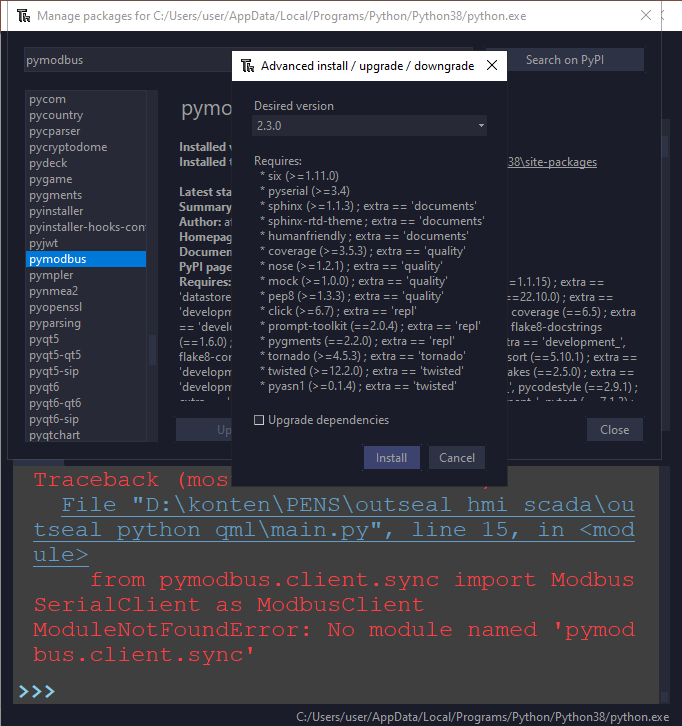
## Instalasi Library Python

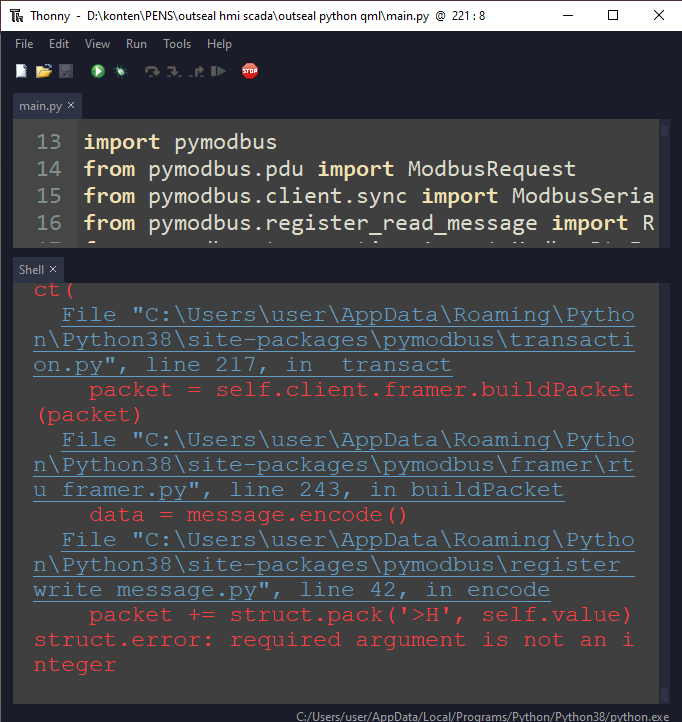
### Install PyQt5

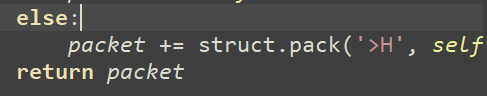


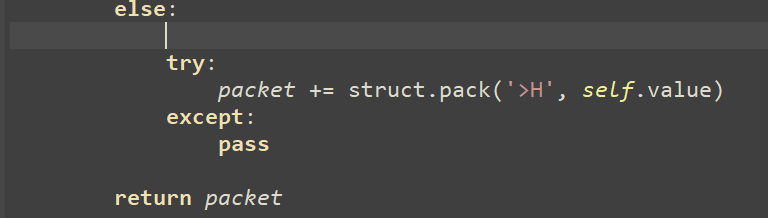
### Install Pymodbus



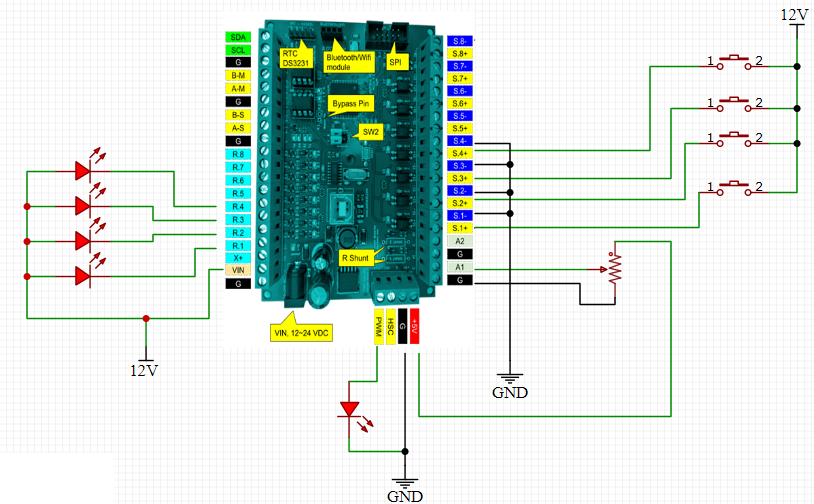








## Wiring Hardware



# Jobsheet 0 : Base Code

## Program Python

###### PROGRAM MEMANGGIL WINDOWS PYQT5 ##########################

####### memanggil library PyQt5 untuk Graphical User Interface ###

#----------------------------------------------------------------#

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5.QtQml import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtQuick import \*

import sys

import threading

#----------------------------------------------------------------#

#########memanggil Library modbus################################

import pymodbus

from pymodbus.pdu import ModbusRequest

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

from pymodbus.register\_read\_message import ReadInputRegistersResponse

from pymodbus.transaction import ModbusRtuFramer

import serial

import time

#---------------DEKLARASI VARIABEL----------------------------#

#---------------SCAN AVAILABLE MODBUS PORT -------------------#

def serial\_ports():

if sys.platform.startswith('win'):

ports = ['COM%s' % (i + 1) for i in range(256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

# this excludes your current terminal "/dev/tty"

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'):

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Unsupported platform')

result = []

for port in ports:

try:

s = serial.Serial(port)

s.close()

result.append(port)

except (OSError, serial.SerialException):

pass

return result

print(str(serial\_ports()))

port = input("modbus port : ")

baudrate = input("modbus baudrate : ")

slave\_id = input("slave id : ")

client = ModbusClient(method='rtu', port=port, baudrate=int(baudrate), parity='N', timeout=4,strict=False)

connection = client.connect()

########## mengisi class table dengan instruksi pyqt5#############

#----------------------------------------------------------------#

class table(QObject):

def \_\_init\_\_(self, parent = None):

super().\_\_init\_\_(parent)

self.app = QApplication(sys.argv)

self.engine = QQmlApplicationEngine(self)

self.engine.rootContext().setContextProperty("backend", self)

self.engine.load(QUrl("main.qml"))

sys.exit(self.app.exec\_())

#----------------------------------------------------------------#

def modbus\_data\_process(num):

while True:

pass

########## memanggil class table di mainloop ######################

#----------------------------------------------------------------#

if \_\_name\_\_ == "\_\_main\_\_":

t1 = threading.Thread(target=modbus\_data\_process, args=(10,))

t1.start()

main = table()

#----------------------------------------------------------------#

## Program QML

import QtQuick 2.12

import QtQuick.Window 2.13

import QtQuick.Controls 2.0

import QtQuick.Controls.Styles 1.4

import QtQuick.Extras 1.4

import QtQuick.Extras.Private 1.0

Window {

id : root

width: 400

//maximumWidth : 1280

//minimumWidth : width

height: 400

//maximumHeight : 800

//minimumHeight : height

title:"membuat MINI SCADA HMI"

color : "#FF96C5"

visible: true

//flags: Qt.WindowMaximized //Qt.Dialog

}

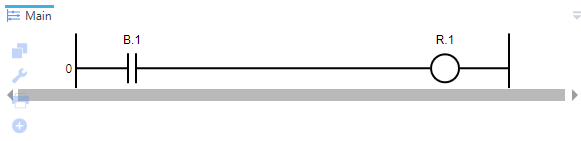
# 

# Jobsheet 1 : Digital Output

## Single Output

### 

### Program Ladder



### Program Python

###### PROGRAM MEMANGGIL WINDOWS PYQT5 ##########################

####### memanggil library PyQt5 untuk Graphical User Interface ##################################

#----------------------------------------------------------------#

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5.QtQml import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtQuick import \*

import sys

import threading

#----------------------------------------------------------------#

import pymodbus

from pymodbus.pdu import ModbusRequest

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

from pymodbus.register\_read\_message import ReadInputRegistersResponse

from pymodbus.transaction import ModbusRtuFramer

import serial

import time

#---------------DEKLARASI VARIABEL----------------------------#

button1\_status = 0

#---------------SCAN AVAILABLE MODBUS PORT -------------------#

def serial\_ports():

if sys.platform.startswith('win'):

ports = ['COM%s' % (i + 1) for i in range(256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

# this excludes your current terminal "/dev/tty"

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'):

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Unsupported platform')

result = []

for port in ports:

try:

s = serial.Serial(port)

s.close()

result.append(port)

except (OSError, serial.SerialException):

pass

return result

print(str(serial\_ports()))

port = input("modbus port : ")

baudrate = input("modbus baudrate : ")

slave\_id = input("slave id : ")

client = ModbusClient(method='rtu', port=port, baudrate=int(baudrate), parity='N', timeout=4,strict=False)

connection = client.connect()

########## mengisi class table dengan instruksi pyqt5#############

#----------------------------------------------------------------#

class table(QObject):

def \_\_init\_\_(self, parent = None):

super().\_\_init\_\_(parent)

self.app = QApplication(sys.argv)

self.engine = QQmlApplicationEngine(self)

self.engine.rootContext().setContextProperty("backend", self)

self.engine.load(QUrl("main.qml"))

sys.exit(self.app.exec\_())

#PROGRAM UNTUK MENERIMA DATA DARI QML

@pyqtSlot(int)

def button1(self, message):

global button1\_status

print(message)

button1\_status = message

#----------------------------------------------------------------#

#------MEMPROSES DATA MODBUS ------------------------------------#

def modbus\_data\_process(num):

while True:

if (button1\_status == 1):

client.write\_coil(128, True, unit=int(slave\_id))

else:

client.write\_coil(128, False, unit=int(slave\_id))

'''

note : jika button bernilai 1 maka python mengirim nilai true ke alamat modbus 128 alias

B1 untuk menyalakan lampu, begitupun sebaliknya

'''

########## memanggil class table di mainloop######################

#----------------------------------------------------------------#

if \_\_name\_\_ == "\_\_main\_\_":

t1 = threading.Thread(target=modbus\_data\_process, args=(10,))

t1.start()

main = table()

#----------------------------------------------------------------#

### Program QML

import QtQuick 2.12

import QtQuick.Window 2.13

import QtQuick.Controls 2.0

import QtQuick.Controls.Styles 1.4

import QtQuick.Extras 1.4

import QtQuick.Extras.Private 1.0

Window {

id : root

width: 400

//maximumWidth : 1280

//minimumWidth : width

height: 400

//maximumHeight : 800

//minimumHeight : height

title:"membuat MINI SCADA HMI"

color : "#FF96C5"

visible: true

//flags: Qt.WindowMaximized //Qt.Dialog

Button{

id : button1

x : 50

y : 150

width : 90

height : 90

checkable : true

checked : false

Rectangle {

id : button1\_color

width : parent.width

height : parent.height

color : "#340040"

}

Text{

x: 10

y : 10

text : "button1"

//font.family : "Comic Sans"

font.pixelSize : 15

}

}

Timer{

id:guitimer

interval: 200

repeat: true

running: true

onTriggered: {

if (button1.checked == true){

button1\_color.color = "#F2055C"

backend.button1("1")

}

if (button1.checked == false){

button1\_color.color = "#340040"

backend.button1("0")

}

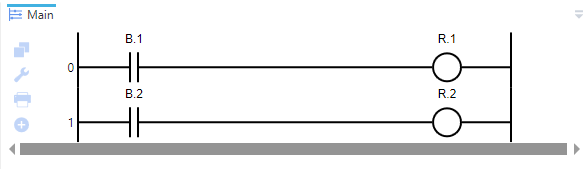
}

}

}

## Multiple Output

### Program Ladder



### Program Python

###### PROGRAM MEMANGGIL WINDOWS PYQT5 ##########################

####### memanggil library PyQt5 untuk Graphical User Interface ##################################

#----------------------------------------------------------------#

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5.QtQml import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtQuick import \*

import sys

import threading

#----------------------------------------------------------------#

import pymodbus

from pymodbus.pdu import ModbusRequest

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

from pymodbus.register\_read\_message import ReadInputRegistersResponse

from pymodbus.transaction import ModbusRtuFramer

import serial

import time

###############DECLARE VARIABLES#############

button1\_status = 0

button2\_status = 0

#---------------SCAN AVAILABLE MODBUS PORT -------------------#

def serial\_ports():

if sys.platform.startswith('win'):

ports = ['COM%s' % (i + 1) for i in range(256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

# this excludes your current terminal "/dev/tty"

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'):

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Unsupported platform')

result = []

for port in ports:

try:

s = serial.Serial(port)

s.close()

result.append(port)

except (OSError, serial.SerialException):

pass

return result

print(str(serial\_ports()))

port = input("modbus port : ")

baudrate = input("modbus baudrate : ")

slave\_id = input("slave id : ")

client = ModbusClient(method='rtu', port=port, baudrate=int(baudrate), parity='N', timeout=4,strict=False)

connection = client.connect()

########## mengisi class table dengan instruksi pyqt5#############

#----------------------------------------------------------------#

class table(QObject):

def \_\_init\_\_(self, parent = None):

super().\_\_init\_\_(parent)

self.app = QApplication(sys.argv)

self.engine = QQmlApplicationEngine(self)

self.engine.rootContext().setContextProperty("backend", self)

self.engine.load(QUrl("main.qml"))

sys.exit(self.app.exec\_())

@pyqtSlot(int)

def button1(self, message):

global button1\_status

#print(message)

button1\_status = message

@pyqtSlot(int)

def button2(self, message):

global button2\_status

#print(message)

button2\_status = message

#----------------------------------------------------------------#

def modbus\_data\_process(num):

while True:

if (button1\_status == 1):

client.write\_coil(128, True, unit=int(slave\_id))

else:

client.write\_coil(128, False, unit=int(slave\_id))

if (button2\_status == 1):

client.write\_coil(129, True, unit=int(slave\_id))

else:

client.write\_coil(129, False, unit=int(slave\_id))

########## memanggil class table di mainloop######################

#----------------------------------------------------------------#

if \_\_name\_\_ == "\_\_main\_\_":

t1 = threading.Thread(target=modbus\_data\_process, args=(10,))

t1.start()

main = table()

#----------------------------------------------------------------#

### Program QML

import QtQuick 2.12

import QtQuick.Window 2.13

import QtQuick.Controls 2.0

import QtQuick.Controls.Styles 1.4

import QtQuick.Extras 1.4

import QtQuick.Extras.Private 1.0

Window {

id : root

width: 400

//maximumWidth : 1280

//minimumWidth : width

height: 400

//maximumHeight : 800

//minimumHeight : height

title:"membuat MINI SCADA HMI"

color : "#FF96C5"

visible: true

//flags: Qt.WindowMaximized //Qt.Dialog

Button{

id : button1

x : 50

y : 150

width : 90

height : 90

checkable : true

checked : false

Rectangle {

id : button1\_color

width : parent.width

height : parent.height

color : "#340040"

}

Text{

x: 10

y : 10

text : "button1"

//font.family : "Comic Sans"

font.pixelSize : 15

}

}

Button{

id : button2

x : 50

y : 250

width : 90

height : 90

checkable : true

checked : false

Rectangle {

id : button2\_color

width : parent.width

height : parent.height

color : "#340040"

}

Text{

x: 10

y : 10

text : "button2"

//font.family : "Comic Sans"

font.pixelSize : 15

}

}

Timer{

id:guitimer

interval: 200

repeat: true

running: true

onTriggered: {

if (button1.checked == true){

button1\_color.color = "#F2055C"

backend.button1("1")

}

if (button1.checked == false){

button1\_color.color = "#340040"

backend.button1("0")

}

if (button2.checked == true){

button2\_color.color = "#F2055C"

backend.button2("1")

}

if (button2.checked == false){

button2\_color.color = "#340040"

backend.button2("0")

}

}

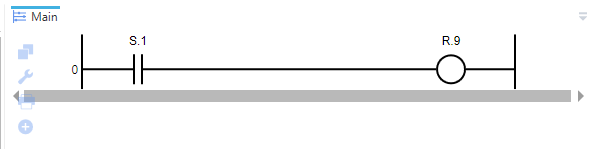
}

}

# Jobsheet 2 : Digital Input

## Single Input

### Program Ladder



### Program Python

###### PROGRAM MEMANGGIL WINDOWS PYQT5 ##########################

####### memanggil library PyQt5 untuk Graphical User Interface ##################################

#----------------------------------------------------------------#

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5.QtQml import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtQuick import \*

import sys

import threading

#----------------------------------------------------------------#

import pymodbus

from pymodbus.pdu import ModbusRequest

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

from pymodbus.register\_read\_message import ReadInputRegistersResponse

from pymodbus.transaction import ModbusRtuFramer

import serial

import time

###############DECLARE VARIABLES#############

indicator1\_status = ""

#---------------SCAN AVAILABLE MODBUS PORT -------------------#

def serial\_ports():

if sys.platform.startswith('win'):

ports = ['COM%s' % (i + 1) for i in range(256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

# this excludes your current terminal "/dev/tty"

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'):

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Unsupported platform')

result = []

for port in ports:

try:

s = serial.Serial(port)

s.close()

result.append(port)

except (OSError, serial.SerialException):

pass

return result

print(str(serial\_ports()))

port = input("modbus port : ")

baudrate = input("modbus baudrate : ")

slave\_id = input("slave id : ")

client = ModbusClient(method='rtu', port=port, baudrate=int(baudrate), parity='N', timeout=4,strict=False)

connection = client.connect()

########## mengisi class table dengan instruksi pyqt5#############

#----------------------------------------------------------------#

class table(QObject):

def \_\_init\_\_(self, parent = None):

super().\_\_init\_\_(parent)

self.app = QApplication(sys.argv)

self.engine = QQmlApplicationEngine(self)

self.engine.rootContext().setContextProperty("backend", self)

self.engine.load(QUrl("main.qml"))

sys.exit(self.app.exec\_())

@pyqtSlot(result=str)

def indicator1\_status(self): return indicator1\_status

#----------------------------------------------------------------#

def modbus\_data\_process(num):

global request

global request\_coil

global indicator1\_status

while True:

request\_coil = client.read\_coils(address=0,count=15,unit=int(slave\_id))

request = client.read\_holding\_registers(address=0,count=0x7,unit=int(slave\_id))

try:

print(request.registers)

#print(request\_coil.bits[8], request\_coil.bits[9], request\_coil.bits[10], request\_coil.bits[11])

if (request\_coil.bits[8] == 0):

indicator1\_status = "off"

else:

indicator1\_status = "on"

except:

pass

########## memanggil class table di mainloop######################

#----------------------------------------------------------------#

if \_\_name\_\_ == "\_\_main\_\_":

t1 = threading.Thread(target=modbus\_data\_process, args=(10,))

t1.start()

main = table()

#----------------------------------------------------------------#

### Program QML

import QtQuick 2.12

import QtQuick.Window 2.13

import QtQuick.Controls 2.0

import QtQuick.Controls.Styles 1.4

import QtQuick.Extras 1.4

import QtQuick.Extras.Private 1.0

Window {

id : root

width: 400

//maximumWidth : 1280

//minimumWidth : width

height: 400

//maximumHeight : 800

//minimumHeight : height

title:"membuat MINI SCADA HMI"

color : "#FF96C5"

visible: true

//flags: Qt.WindowMaximized //Qt.Dialog

Rectangle {

id : indicator1\_color

x: 50

y : 100

width : 250

height : 40

color : "#340040"

Text {

id : indicator1

anchors.horizontalCenter: parent.horizontalCenter

anchors.verticalCenter: parent.verticalCenter

font.family : "HarmoniaSansW01-Bold"

font.pixelSize : 20

text : "R9"

color : "#07F2F2"

}

}

Timer{

id:guitimer

interval: 200

repeat: true

running: true

onTriggered: {

if (backend.indicator1\_status() == "off"){

indicator1.text = "R9 : 0"

indicator1\_color.color = "#340040"

}

if (backend.indicator1\_status() == "on"){

indicator1.text = "R9 : 1"

indicator1\_color.color = "#F2055C"

}

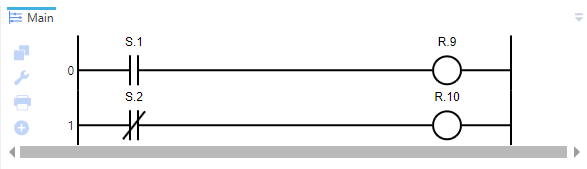
}

}

}

## Multiple Input

### Program Ladder



### Program Python

###### PROGRAM MEMANGGIL WINDOWS PYQT5 ##########################

####### memanggil library PyQt5 untuk Graphical User Interface ##################################

#----------------------------------------------------------------#

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5.QtQml import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtQuick import \*

import sys

import threading

#----------------------------------------------------------------#

import pymodbus

from pymodbus.pdu import ModbusRequest

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

from pymodbus.register\_read\_message import ReadInputRegistersResponse

from pymodbus.transaction import ModbusRtuFramer

import serial

import time

###############DECLARE VARIABLES#############

indicator1\_status = ""

indicator2\_status = ""

#---------------SCAN AVAILABLE MODBUS PORT -------------------#

def serial\_ports():

if sys.platform.startswith('win'):

ports = ['COM%s' % (i + 1) for i in range(256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

# this excludes your current terminal "/dev/tty"

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'):

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Unsupported platform')

result = []

for port in ports:

try:

s = serial.Serial(port)

s.close()

result.append(port)

except (OSError, serial.SerialException):

pass

return result

print(str(serial\_ports()))

port = input("modbus port : ")

baudrate = input("modbus baudrate : ")

slave\_id = input("slave id : ")

client = ModbusClient(method='rtu', port=port, baudrate=int(baudrate), parity='N', timeout=4,strict=False)

connection = client.connect()

########## mengisi class table dengan instruksi pyqt5#############

#----------------------------------------------------------------#

class table(QObject):

def \_\_init\_\_(self, parent = None):

super().\_\_init\_\_(parent)

self.app = QApplication(sys.argv)

self.engine = QQmlApplicationEngine(self)

self.engine.rootContext().setContextProperty("backend", self)

self.engine.load(QUrl("main.qml"))

sys.exit(self.app.exec\_())

@pyqtSlot(result=str)

def indicator1\_status(self): return indicator1\_status

@pyqtSlot(result=str)

def indicator2\_status(self): return indicator2\_status

#----------------------------------------------------------------#

def modbus\_data\_process(num):

global request

global request\_coil

global indicator1\_status

global indicator2\_status

while True:

request\_coil = client.read\_coils(address=0,count=15,unit=int(slave\_id))

request = client.read\_holding\_registers(address=0,count=0x7,unit=int(slave\_id))

try:

print(request.registers)

#print(request\_coil.bits[8], request\_coil.bits[9], request\_coil.bits[10], request\_coil.bits[11])

if (request\_coil.bits[8] == 0):

indicator1\_status = "off"

else:

indicator1\_status = "on"

if (request\_coil.bits[9] == 0):

indicator2\_status = "off"

else:

indicator2\_status = "on"

except:

pass

########## memanggil class table di mainloop######################

#----------------------------------------------------------------#

if \_\_name\_\_ == "\_\_main\_\_":

t1 = threading.Thread(target=modbus\_data\_process, args=(10,))

t1.start()

main = table()

#----------------------------------------------------------------#

### Program QML

import QtQuick 2.12

import QtQuick.Window 2.13

import QtQuick.Controls 2.0

import QtQuick.Controls.Styles 1.4

import QtQuick.Extras 1.4

import QtQuick.Extras.Private 1.0

Window {

id : root

width: 400

//maximumWidth : 1280

//minimumWidth : width

height: 400

//maximumHeight : 800

//minimumHeight : height

title:"membuat MINI SCADA HMI"

color : "#FF96C5"

visible: true

//flags: Qt.WindowMaximized //Qt.Dialog

Rectangle {

id : indicator1\_color

x: 50

y : 100

width : 250

height : 40

color : "#340040"

Text {

id : indicator1

anchors.horizontalCenter: parent.horizontalCenter

anchors.verticalCenter: parent.verticalCenter

font.family : "HarmoniaSansW01-Bold"

font.pixelSize : 20

text : "R9"

color : "#07F2F2"

}

}

Rectangle {

id : indicator2\_color

x: 50

y : 160

width : 250

height : 40

color : "#340040"

Text {

id : indicator2

anchors.horizontalCenter: parent.horizontalCenter

anchors.verticalCenter: parent.verticalCenter

font.family : "HarmoniaSansW01-Bold"

font.pixelSize : 20

text : "R10"

color : "#07F2F2"

}

}

Timer{

id:guitimer

interval: 200

repeat: true

running: true

onTriggered: {

if (backend.indicator1\_status() == "off"){

indicator1.text = "R9 : 0"

indicator1\_color.color = "#340040"

}

if (backend.indicator1\_status() == "on"){

indicator1.text = "R9 : 1"

indicator1\_color.color = "#F2055C"

}

if (backend.indicator2\_status() == "off"){

indicator2.text = "R10 : 0"

indicator2\_color.color = "#340040"

}

if (backend.indicator2\_status() == "on"){

indicator2.text = "R10 : 1"

indicator2\_color.color = "#F2055C"

}

}

}

}

# Jobsheet 3 : Analog Input

## Basic

### Program Ladder



### Program Python

###### PROGRAM MEMANGGIL WINDOWS PYQT5 ##########################

####### memanggil library PyQt5 untuk Graphical User Interface ##################################

#----------------------------------------------------------------#

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5.QtQml import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtQuick import \*

import sys

import threading

#----------------------------------------------------------------#

import pymodbus

from pymodbus.pdu import ModbusRequest

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

from pymodbus.register\_read\_message import ReadInputRegistersResponse

from pymodbus.transaction import ModbusRtuFramer

import serial

import time

##########################################

analog = 0

#---------------SCAN AVAILABLE MODBUS PORT -------------------#

def serial\_ports():

if sys.platform.startswith('win'):

ports = ['COM%s' % (i + 1) for i in range(256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

# this excludes your current terminal "/dev/tty"

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'):

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Unsupported platform')

result = []

for port in ports:

try:

s = serial.Serial(port)

s.close()

result.append(port)

except (OSError, serial.SerialException):

pass

return result

print(str(serial\_ports()))

port = input("modbus port : ")

baudrate = input("modbus baudrate : ")

slave\_id = input("slave id : ")

client = ModbusClient(method='rtu', port=port, baudrate=int(baudrate), parity='N', timeout=4,strict=False)

connection = client.connect()

########## mengisi class table dengan instruksi pyqt5#############

#----------------------------------------------------------------#

class table(QObject):

def \_\_init\_\_(self, parent = None):

super().\_\_init\_\_(parent)

self.app = QApplication(sys.argv)

self.engine = QQmlApplicationEngine(self)

self.engine.rootContext().setContextProperty("backend", self)

self.engine.load(QUrl("main.qml"))

sys.exit(self.app.exec\_())

@pyqtSlot(result=int)

def sensor(self): return analog

#----------------------------------------------------------------#

def modbus\_data\_process(num):

global request

global request\_coil

global analog

while True:

request = client.read\_holding\_registers(address=0,count=0x7,unit=int(slave\_id))

request\_coil = client.read\_coils(address=0,count=15,unit=int(slave\_id))

try:

analog = request.registers[0]

except:

pass

########## memanggil class table di mainloop######################

#----------------------------------------------------------------#

if \_\_name\_\_ == "\_\_main\_\_":

t1 = threading.Thread(target=modbus\_data\_process, args=(10,))

t1.start()

main = table()

#----------------------------------------------------------------#

### Program QML

import QtQuick 2.12

import QtQuick.Window 2.13

import QtQuick.Controls 2.0

import QtQuick.Controls.Styles 1.4

import QtQuick.Extras 1.4

import QtQuick.Extras.Private 1.0

Window {

id : root

width: 400

//maximumWidth : 1280

//minimumWidth : width

height: 400

//maximumHeight : 800

//minimumHeight : height

title:"membuat MINI SCADA HMI"

color : "black"

visible: true

//flags: Qt.WindowMaximized //Qt.Dialog

CircularGauge {

id : gauge

x: 10

y: 70

height : 250

width : 250

value: 0

minimumValue: 0

maximumValue: 100

style: CircularGaugeStyle {

labelStepSize: 10

}

}

Timer{

id:tmgauge

interval: 200

repeat: true

running: true

onTriggered: {

gauge.value = backend.sensor()

}

}

}

## Customize

### Program Ladder



### Program Python

###### PROGRAM MEMANGGIL WINDOWS PYQT5 ##########################

####### memanggil library PyQt5 untuk Graphical User Interface ##################################

#----------------------------------------------------------------#

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5.QtQml import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtQuick import \*

import sys

import threading

#----------------------------------------------------------------#

import pymodbus

from pymodbus.pdu import ModbusRequest

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

from pymodbus.register\_read\_message import ReadInputRegistersResponse

from pymodbus.transaction import ModbusRtuFramer

import serial

import time

##########################################

analog = 0

#---------------SCAN AVAILABLE MODBUS PORT -------------------#

def serial\_ports():

if sys.platform.startswith('win'):

ports = ['COM%s' % (i + 1) for i in range(256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

# this excludes your current terminal "/dev/tty"

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'):

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Unsupported platform')

result = []

for port in ports:

try:

s = serial.Serial(port)

s.close()

result.append(port)

except (OSError, serial.SerialException):

pass

return result

print(str(serial\_ports()))

port = input("modbus port : ")

baudrate = input("modbus baudrate : ")

slave\_id = input("slave id : ")

client = ModbusClient(method='rtu', port=port, baudrate=int(baudrate), parity='N', timeout=4,strict=False)

connection = client.connect()

########## mengisi class table dengan instruksi pyqt5#############

#----------------------------------------------------------------#

class table(QObject):

def \_\_init\_\_(self, parent = None):

super().\_\_init\_\_(parent)

self.app = QApplication(sys.argv)

self.engine = QQmlApplicationEngine(self)

self.engine.rootContext().setContextProperty("backend", self)

self.engine.load(QUrl("main.qml"))

sys.exit(self.app.exec\_())

@pyqtSlot(result=int)

def sensor(self): return analog

#----------------------------------------------------------------#

def modbus\_data\_process(num):

global request

global request\_coil

global analog

while True:

request = client.read\_holding\_registers(address=0,count=0x7,unit=int(slave\_id))

request\_coil = client.read\_coils(address=0,count=15,unit=int(slave\_id))

try:

analog = request.registers[0]

except:

pass

########## memanggil class table di mainloop######################

#----------------------------------------------------------------#

if \_\_name\_\_ == "\_\_main\_\_":

t1 = threading.Thread(target=modbus\_data\_process, args=(10,))

t1.start()

main = table()

#----------------------------------------------------------------#

### Program QML

import QtQuick 2.12

import QtQuick.Window 2.13

import QtQuick.Controls 2.0

import QtQuick.Controls.Styles 1.4

import QtQuick.Extras 1.4

import QtQuick.Extras.Private 1.0

import "controls"

Window {

id : root

width: 800

//maximumWidth : 1280

//minimumWidth : width

height: 400

//maximumHeight : 800

//minimumHeight : height

title:"membuat MINI SCADA HMI"

color : "black"

visible: true

//flags: Qt.WindowMaximized //Qt.Dialog

CircularSlider {

id: gauge

hideProgress: true

hideTrack: true

width: 300

height: 300

interactive: false

value: 75

minValue: 0

maxValue: 100

Repeater {

model: 72

Rectangle {

id: indicator

width: 5

height: 20

radius: width / 2

color: indicator.angle > gauge.angle ? "#16171C" : "#F2055C"

readonly property real angle: index \* 5

transform: [

Translate {

x: gauge.width / 2 - width / 2

},

Rotation {

origin.x: gauge.width / 2

origin.y: gauge.height / 2

angle: indicator.angle

}

]

}

}

Text{

anchors.horizontalCenter: parent.horizontalCenter

anchors.verticalCenter: parent.verticalCenter

text : gauge.value

font.pixelSize : 30

color : "white"

}

}

CircularSlider {

id: gauge2

x: 350

y: 0

value: 0

//onValueChanged: handlePage.newVal = value

interactive: false

width: 300

height: 300

startAngle: 40

endAngle: 320

rotation: 180

trackWidth: 5

progressWidth: 20

minValue: 0

maxValue: 100

progressColor: "#F2055C"

capStyle: Qt.FlatCap

handle: Rectangle {

transform: Translate {

x: (gauge2.handleWidth - width) / 2

y: gauge2.handleHeight / 2

}

width: 10

height: gauge2.height / 2

color: "#05F26C"

radius: width / 2

antialiasing: true

border.width : 3

border.color : "#340040"

}

Text {

anchors.centerIn: parent

anchors.verticalCenterOffset: -40

rotation: 180

font.pointSize: 20

color: "white"

text: Number(gauge2.value).toFixed()

//font.family : "HarmoniaSansW01-Bold"

}

}

Timer{

id:tmgauge

interval: 200

repeat: true

running: true

onTriggered: {

gauge.value = backend.sensor()

gauge2.value = backend.sensor()

}

}

# Jobsheet 4 : Analog Output

## Basic

### Program Ladder



### Program Python

###### PROGRAM MEMANGGIL WINDOWS PYQT5 ##########################

####### memanggil library PyQt5 untuk Graphical User Interface ##################################

#----------------------------------------------------------------#

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5.QtQml import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtQuick import \*

import sys

import threading

#----------------------------------------------------------------#

import pymodbus

from pymodbus.pdu import ModbusRequest

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

from pymodbus.register\_read\_message import ReadInputRegistersResponse

from pymodbus.transaction import ModbusRtuFramer

import serial

import time

##########################################

slider = ""

#---------------SCAN AVAILABLE MODBUS PORT -------------------#

def serial\_ports():

if sys.platform.startswith('win'):

ports = ['COM%s' % (i + 1) for i in range(256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

# this excludes your current terminal "/dev/tty"

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'):

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Unsupported platform')

result = []

for port in ports:

try:

s = serial.Serial(port)

s.close()

result.append(port)

except (OSError, serial.SerialException):

pass

return result

print(str(serial\_ports()))

port = input("modbus port : ")

baudrate = input("modbus baudrate : ")

slave\_id = input("slave id : ")

client = ModbusClient(method='rtu', port=port, baudrate=int(baudrate), parity='N', timeout=4,strict=False)

connection = client.connect()

########## mengisi class table dengan instruksi pyqt5#############

#----------------------------------------------------------------#

class table(QObject):

def \_\_init\_\_(self, parent = None):

super().\_\_init\_\_(parent)

self.app = QApplication(sys.argv)

self.engine = QQmlApplicationEngine(self)

self.engine.rootContext().setContextProperty("backend", self)

self.engine.load(QUrl("main.qml"))

sys.exit(self.app.exec\_())

@pyqtSlot(int)

def slider(self, message):

#print(message)

global slider

slider = message

#----------------------------------------------------------------#

def modbus\_data\_process(num):

global request

global request\_coil

global analog

while True:

request = client.read\_holding\_registers(address=0,count=0x7,unit=int(slave\_id))

request\_coil = client.read\_coils(address=0,count=15,unit=int(slave\_id))

try:

client.write\_register(0, slider\*10 ,unit=int(slave\_id))

except:

pass

########## memanggil class table di mainloop######################

#----------------------------------------------------------------#

if \_\_name\_\_ == "\_\_main\_\_":

t1 = threading.Thread(target=modbus\_data\_process, args=(10,))

t1.start()

main = table()

#----------------------------------------------------------------#

### Program QML

import QtQuick 2.12

import QtQuick.Window 2.13

import QtQuick.Controls 2.0

import QtQuick.Controls.Styles 1.4

import QtQuick.Extras 1.4

import QtQuick.Extras.Private 1.0

Window {

id : root

width: 400

//maximumWidth : 1280

//minimumWidth : width

height: 400

//maximumHeight : 800

//minimumHeight : height

title:"membuat MINI SCADA HMI"

color : "black"

visible: true

//flags: Qt.WindowMaximized //Qt.Dialog

Slider {

id : slider

x: 10

y: 70

height : 250

width : 250

value: 0

from: 0

to: 100

}

Timer{

id:tmgauge

interval: 200

repeat: true

running: true

onTriggered: {

backend.slider(slider.value)

}

}

}

## Customize

### Program Ladder



### Program Python

###### PROGRAM MEMANGGIL WINDOWS PYQT5 ##########################

####### memanggil library PyQt5 untuk Graphical User Interface ##################################

#----------------------------------------------------------------#

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5.QtQml import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtQuick import \*

import sys

import threading

#----------------------------------------------------------------#

import pymodbus

from pymodbus.pdu import ModbusRequest

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

from pymodbus.register\_read\_message import ReadInputRegistersResponse

from pymodbus.transaction import ModbusRtuFramer

import serial

import time

##########################################

slider = ""

#---------------SCAN AVAILABLE MODBUS PORT -------------------#

def serial\_ports():

if sys.platform.startswith('win'):

ports = ['COM%s' % (i + 1) for i in range(256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

# this excludes your current terminal "/dev/tty"

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'):

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Unsupported platform')

result = []

for port in ports:

try:

s = serial.Serial(port)

s.close()

result.append(port)

except (OSError, serial.SerialException):

pass

return result

print(str(serial\_ports()))

port = input("modbus port : ")

baudrate = input("modbus baudrate : ")

slave\_id = input("slave id : ")

client = ModbusClient(method='rtu', port=port, baudrate=int(baudrate), parity='N', timeout=4,strict=False)

connection = client.connect()

########## mengisi class table dengan instruksi pyqt5#############

#----------------------------------------------------------------#

class table(QObject):

def \_\_init\_\_(self, parent = None):

super().\_\_init\_\_(parent)

self.app = QApplication(sys.argv)

self.engine = QQmlApplicationEngine(self)

self.engine.rootContext().setContextProperty("backend", self)

self.engine.load(QUrl("main.qml"))

sys.exit(self.app.exec\_())

@pyqtSlot(int)

def slider(self, message):

#print(message)

global slider

slider = message

#----------------------------------------------------------------#

def modbus\_data\_process(num):

global request

global request\_coil

global analog

while True:

request = client.read\_holding\_registers(address=0,count=0x7,unit=int(slave\_id))

request\_coil = client.read\_coils(address=0,count=15,unit=int(slave\_id))

try:

client.write\_register(0, slider\*10 ,unit=int(slave\_id))

except:

pass

########## memanggil class table di mainloop######################

#----------------------------------------------------------------#

if \_\_name\_\_ == "\_\_main\_\_":

t1 = threading.Thread(target=modbus\_data\_process, args=(10,))

t1.start()

main = table()

#----------------------------------------------------------------#

### Program QML

import QtQuick 2.12

import QtQuick.Window 2.13

import QtQuick.Controls 2.0

import QtQuick.Controls.Styles 1.4

import QtQuick.Extras 1.4

import QtQuick.Extras.Private 1.0

import "controls"

Window {

id : root

width: 400

//maximumWidth : 1280

//minimumWidth : width

height: 400

//maximumHeight : 800

//minimumHeight : height

title:"membuat MINI SCADA HMI"

color : "black"

visible: true

//flags: Qt.WindowMaximized //Qt.Dialog

CircularSlider {

id: gauge

hideProgress: true

hideTrack: true

width: 300

height: 300

interactive: true

value: 0//(gauge2.value).toFixed()

minValue: 0

maxValue: 100

Repeater {

model: 72

Rectangle {

id: indicator

width: 5

height: 20

radius: width / 2

color: indicator.angle > gauge.angle ? "#16171C" : "#F2055C"

readonly property real angle: index \* 5

transform: [

Translate {

x: gauge.width / 2 - width / 2

},

Rotation {

origin.x: gauge.width / 2

origin.y: gauge.height / 2

angle: indicator.angle

}

]

}

}

Text{

anchors.horizontalCenter: parent.horizontalCenter

anchors.verticalCenter: parent.verticalCenter

text : gauge.value

font.pixelSize : 30

color : "white"

}

}

CircularSlider {

id: gauge2

x: 350

y: 0

value: 0//gauge.value

//onValueChanged: handlePage.newVal = value

interactive: true

width: 300

height: 300

startAngle: 40

endAngle: 320

rotation: 180

trackWidth: 5

progressWidth: 20

minValue: 0

maxValue: 100

progressColor: "#F2055C"

capStyle: Qt.FlatCap

handle: Rectangle {

transform: Translate {

x: (gauge2.handleWidth - width) / 2

y: gauge2.handleHeight / 2

}

width: 10

height: gauge2.height / 2

color: "#05F26C"

radius: width / 2

antialiasing: true

border.width : 3

border.color : "#340040"

}

Text {

anchors.centerIn: parent

anchors.verticalCenterOffset: -40

rotation: 180

font.pointSize: 20

color: "white"

text: Number(gauge2.value).toFixed()

//font.family : "HarmoniaSansW01-Bold"

}

}

Timer{

id:tmgauge

interval: 200

repeat: true

running: true

onTriggered: {

backend.slider(gauge2.value)

gauge.value = (gauge2.value).toFixed()

}

}

}

# Jobsheet 5 : Historical Graph

## Program Ladder



## Program Python

###### PROGRAM MEMANGGIL WINDOWS PYQT5 ##########################

####### memanggil library PyQt5 untuk Graphical User Interface ##################################

#----------------------------------------------------------------#

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5.QtQml import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtQuick import \*

import sys

import threading

#----------------------------------------------------------------#

import pymodbus

from pymodbus.pdu import ModbusRequest

from pymodbus.client.sync import ModbusSerialClient as ModbusClient

from pymodbus.register\_read\_message import ReadInputRegistersResponse

from pymodbus.transaction import ModbusRtuFramer

import serial

import time

##########################################

analog = 0

#---------------SCAN AVAILABLE MODBUS PORT -------------------#

def serial\_ports():

if sys.platform.startswith('win'):

ports = ['COM%s' % (i + 1) for i in range(256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

# this excludes your current terminal "/dev/tty"

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'):

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Unsupported platform')

result = []

for port in ports:

try:

s = serial.Serial(port)

s.close()

result.append(port)

except (OSError, serial.SerialException):

pass

return result

print(str(serial\_ports()))

port = input("modbus port : ")

baudrate = input("modbus baudrate : ")

slave\_id = input("slave id : ")

client = ModbusClient(method='rtu', port=port, baudrate=int(baudrate), parity='N', timeout=4,strict=False)

connection = client.connect()

########## mengisi class table dengan instruksi pyqt5#############

#----------------------------------------------------------------#

class table(QObject):

def \_\_init\_\_(self, parent = None):

super().\_\_init\_\_(parent)

self.app = QApplication(sys.argv)

self.engine = QQmlApplicationEngine(self)

self.engine.rootContext().setContextProperty("backend", self)

self.engine.load(QUrl("main.qml"))

sys.exit(self.app.exec\_())

@pyqtSlot(result=int)

def get\_tiempo(self):

date\_time = QDateTime.currentDateTime()

unixTIME = date\_time.toSecsSinceEpoch()

#unixTIMEx = date\_time.currentMSecsSinceEpoch()

return unixTIME

@pyqtSlot(result=int)

def sensor(self): return analog

#----------------------------------------------------------------#

def modbus\_data\_process(num):

global request

global request\_coil

global analog

while True:

request = client.read\_holding\_registers(address=0,count=0x7,unit=int(slave\_id))

request\_coil = client.read\_coils(address=0,count=15,unit=int(slave\_id))

try:

analog = request.registers[0]

except:

pass

########## memanggil class table di mainloop######################

#----------------------------------------------------------------#

if \_\_name\_\_ == "\_\_main\_\_":

t1 = threading.Thread(target=modbus\_data\_process, args=(10,))

t1.start()

main = table()

#----------------------------------------------------------------#

## Program QML

import QtQuick 2.12

import QtQuick.Window 2.13

import QtQuick.Controls 2.0

import QtQuick.Controls.Styles 1.4

import QtQuick.Extras 1.4

import QtQuick.Extras.Private 1.0

import QtCharts 2.1

Window {

id : root

width: 400

//maximumWidth : 1280

//minimumWidth : width

height: 400

//maximumHeight : 800

//minimumHeight : height

title:"membuat MINI SCADA HMI"

color : "black"

visible: true

//flags: Qt.WindowMaximized //Qt.Dialog

Rectangle {

x: 0

y: 0

width: parent.width

height: parent.height

color:"#47B5FF"

ChartView {

id : cv

//title: "PAYOUT ROPE"

antialiasing: true

legend.visible: false

height: parent.height

anchors.right: parent.right

anchors.left: parent.left

//theme: ChartView.ChartThemeLight

backgroundColor:"#1089FF"

property int timcnt: 0

property double valueCH1: 0

property double valueCH2: 0

property double valueCH3: 0

property double valueCH4: 0

//property double valueTM1: 0

property double periodGRAPH: 30 // Seconds

property double startTIME: 0

property double intervalTM: 200 // miliseconds

ValueAxis{

id:yAxis1

min: 0

max : 100

tickCount: 1

//labelFormat: "%d"

labelsColor: "yellow"

}

LineSeries {

//name: "LineSeries"

name: "AIN 0"

id:lines1

width: 4

color: "#21209C"

axisY: yAxis1

axisX: DateTimeAxis {

id: eje4

//format: "yyyy MMM"

format:"HH:mm:ss.z"

//format:"mm:ss.z"

}

}

}

}

Timer{

id:tm

interval: cv.intervalTM

repeat: true

running: true

onTriggered: {

cv.timcnt = cv.timcnt + 1

cv.valueCH1 = backend.sensor()

if (lines1.count>cv.periodGRAPH\*100/cv.intervalTM){

lines1.remove(0)

}

lines1.append(cv.startTIME+cv.timcnt\*cv.intervalTM ,cv.valueCH1)

lines1.axisX.min = new Date(cv.startTIME-cv.periodGRAPH\*100 + cv.timcnt\*cv.intervalTM)

lines1.axisX.max = new Date(cv.startTIME + cv.timcnt\*cv.intervalTM)

}

}

Component.onCompleted: {

cv.startTIME = backend.get\_tiempo()\*1000

}

}