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

****

****

Disusun oleh:

Muhammad Husni Muttaqin, S. Pd.

**Hasil Kerjasama antara :**

**Lab. Mekatronika Politeknik Elektronika Negeri Surabaya**

**Komunitas Belajar Arduino dan Mekatronika**

**Komunitas Modbus Indonesia**

**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 saya 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 2](#_Toc125196611)

[Daftar Isi 3](#_Toc125196612)

[Persiapan 5](#_Toc125196613)

[Pengenalan Software Tools 5](#_Toc125196614)

[Notepad++ 5](#_Toc125196615)

[Thonny IDE 6](#_Toc125196616)

[Outseal Studio 8](#_Toc125196617)

[Instalasi Library Python 9](#_Toc125196618)

[Install PyQt5 9](#_Toc125196619)

[Install PyQtChart 10](#_Toc125196620)

[Install Pymodbus 11](#_Toc125196621)

[Wiring Hardware 13](#_Toc125196622)

[Program tampilan dasar pada QML 14](#_Toc125196623)

[1. Membuat Teks 14](#_Toc125196624)

[2. Membuat Button 14](#_Toc125196625)

[3. Membuat Rectangle 15](#_Toc125196626)

[4. Membuat Slider 16](#_Toc125196627)

[5. Membuat Gauge 18](#_Toc125196628)

[Tabel konversi outseal PLC Code ke alamat MODBUS yang digunakan 19](#_Toc125196629)

[Jobsheet 0 : Base Code 20](#_Toc125196630)

[Program Python 20](#_Toc125196631)

[Program QML 23](#_Toc125196632)

[Jobsheet 1 : Digital Output 25](#_Toc125196633)

[Single Output 25](#_Toc125196634)

[Program Ladder 25](#_Toc125196635)

[Program Python 26](#_Toc125196636)

[Program QML 30](#_Toc125196637)

[Multiple Output 33](#_Toc125196638)

[Program Ladder 33](#_Toc125196639)

[Program Python 33](#_Toc125196640)

[Program QML 37](#_Toc125196641)

[Jobsheet 2 : Digital Input 42](#_Toc125196642)

[Single Input 42](#_Toc125196643)

[Program Ladder 42](#_Toc125196644)

[Program Python 43](#_Toc125196645)

[Program QML 47](#_Toc125196646)

[Multiple Input 50](#_Toc125196647)

[Program Ladder 50](#_Toc125196648)

[Program Python 50](#_Toc125196649)

[Program QML 54](#_Toc125196650)

[Jobsheet 3 : Analog Input 59](#_Toc125196651)

[Basic 59](#_Toc125196652)

[Program Ladder 59](#_Toc125196653)

[Program Python 60](#_Toc125196654)

[Program QML 64](#_Toc125196655)

[Customize 66](#_Toc125196656)

[Program Ladder 66](#_Toc125196657)

[Program Python 67](#_Toc125196658)

[Program QML 70](#_Toc125196659)

[Jobsheet 4 : Analog Output 76](#_Toc125196660)

[Basic 76](#_Toc125196661)

[Program Ladder 76](#_Toc125196662)

[Program Python 76](#_Toc125196663)

[Program QML 80](#_Toc125196664)

[Customize 82](#_Toc125196665)

[Program Ladder 82](#_Toc125196666)

[Program Python 82](#_Toc125196667)

[Program QML 86](#_Toc125196668)

[Jobsheet 5 : Historical Graph 92](#_Toc125196669)

[Program Ladder 92](#_Toc125196670)

[Program Python 92](#_Toc125196671)

[Program QML 97](#_Toc125196672)

[Tentang Penulis 101](#_Toc125196673)

# 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:

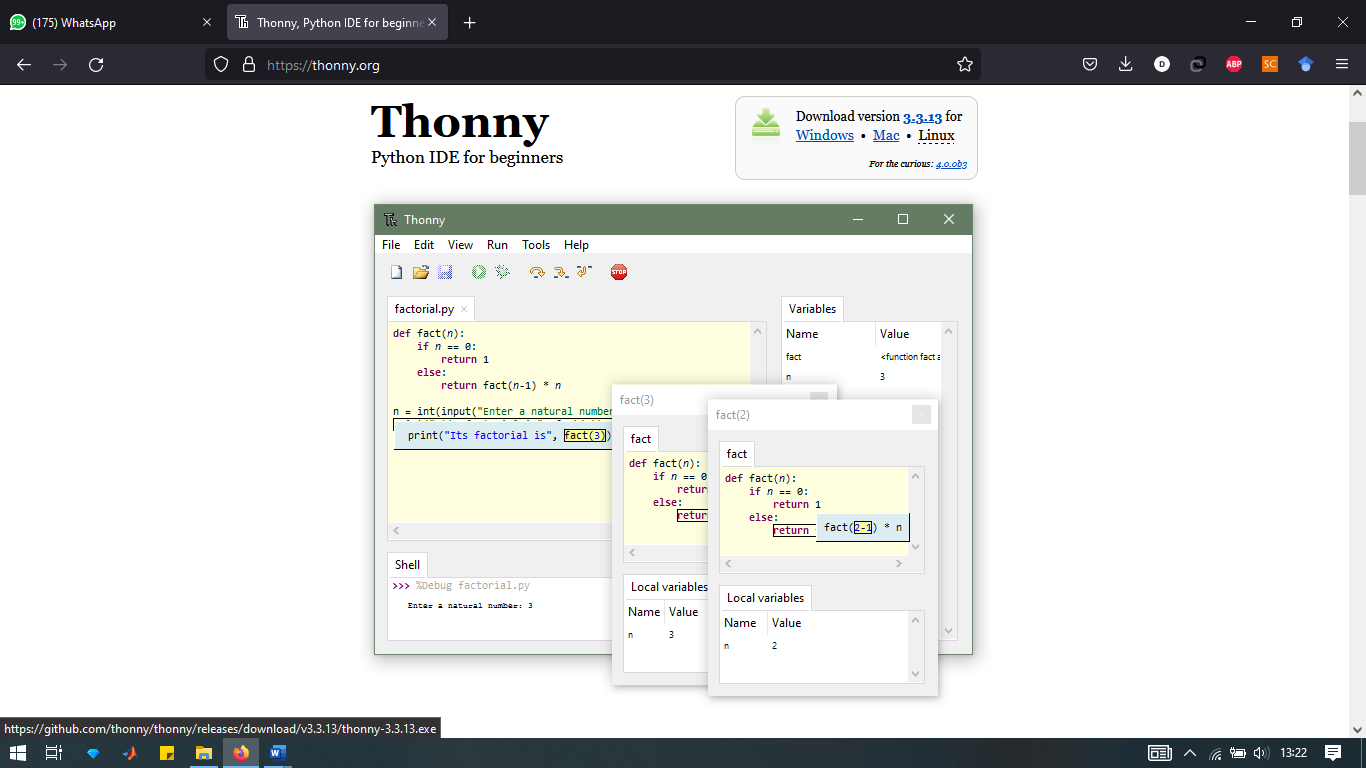
Graphical user interface, application, Word

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

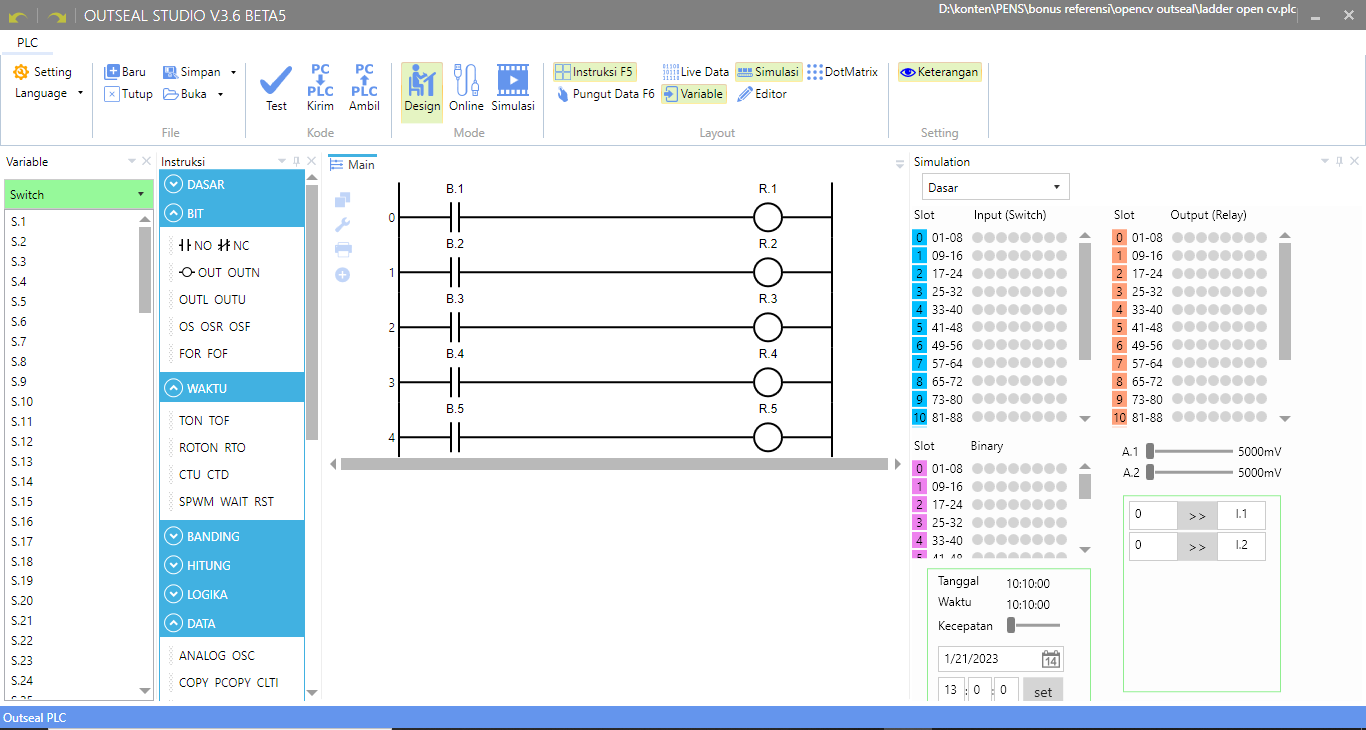
Description automatically generated

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.

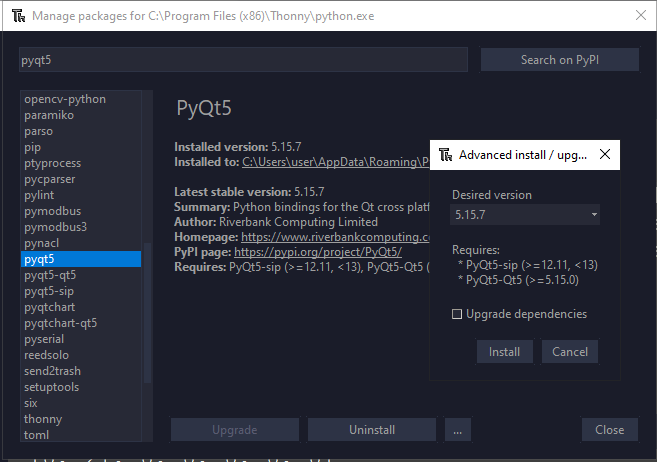
### Outseal Studio

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

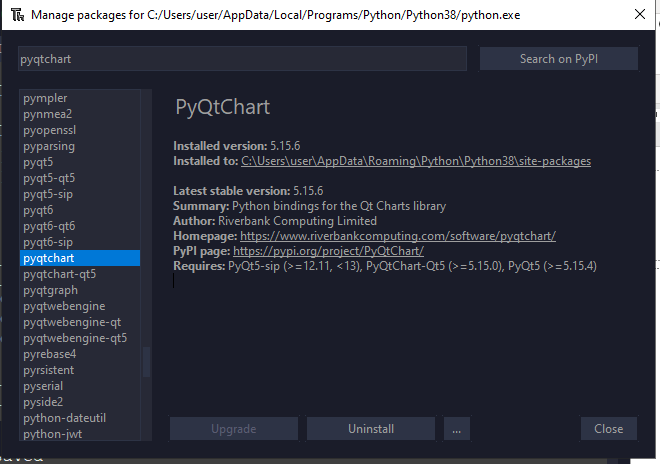
### Install PyQt5



Cara install library PyQt5 adalah sebagai berikut :

1. Klik Tools
2. Klik manage packages
3. lalu ketik PyQt5
4. lalu pilih versi 5.15.7

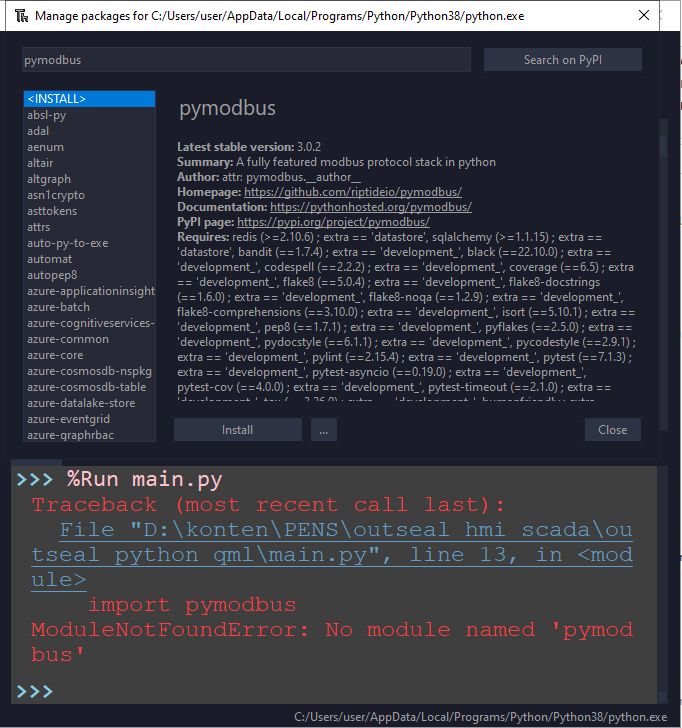
### Install PyQtChart

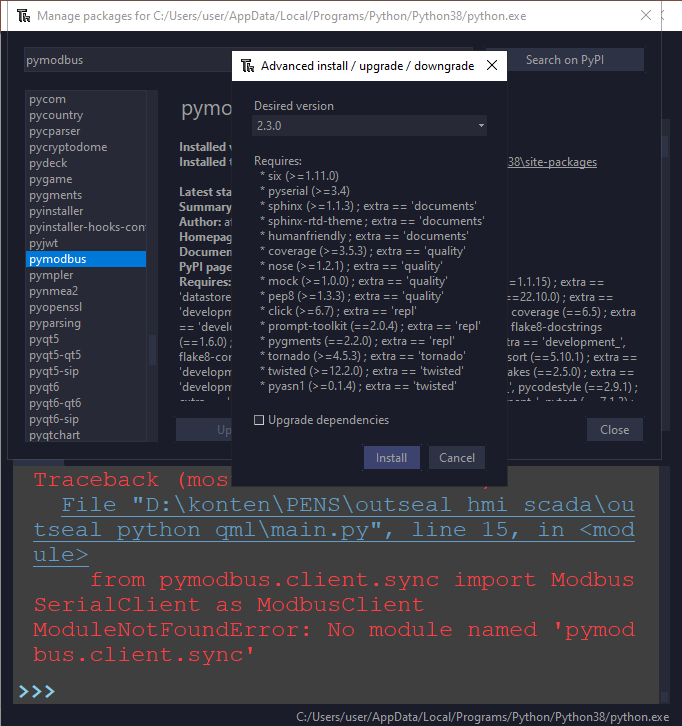


Cara install library PyQtChart adalah sebagai berikut :

1. Klik Tools
2. Klik manage packages
3. lalu ketik PyQtChart
4. lalu pilih versi 5.15.6

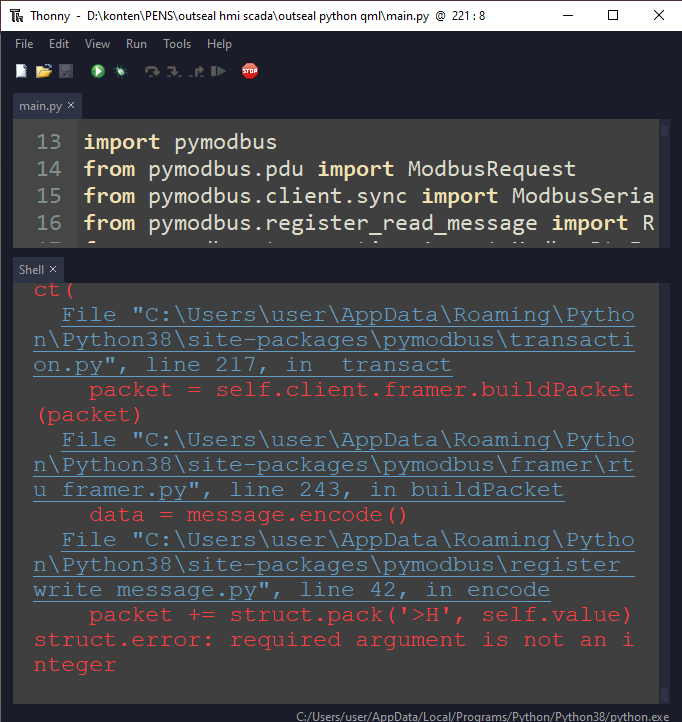
### Install Pymodbus





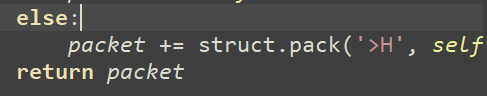
Cara install library PyModbus adalah sebagai berikut :

1. Klik Tools
2. Klik manage packages
3. lalu ketik pymodbus
4. lalu pilih versi 2.3.0
5. Jalankan program yang menggunakan pymodbus, apabila ada error packet cara mengatasinya adalah klik error in encode

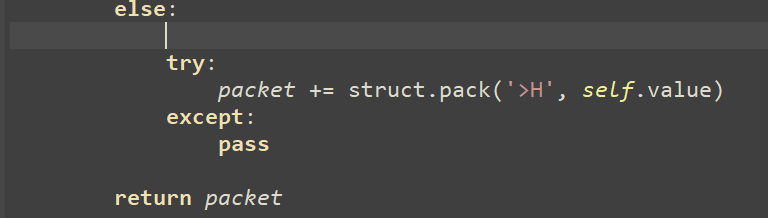


1. Setelah klik error in encode maka tambahkan try and except pada packet seperti dibawah ini

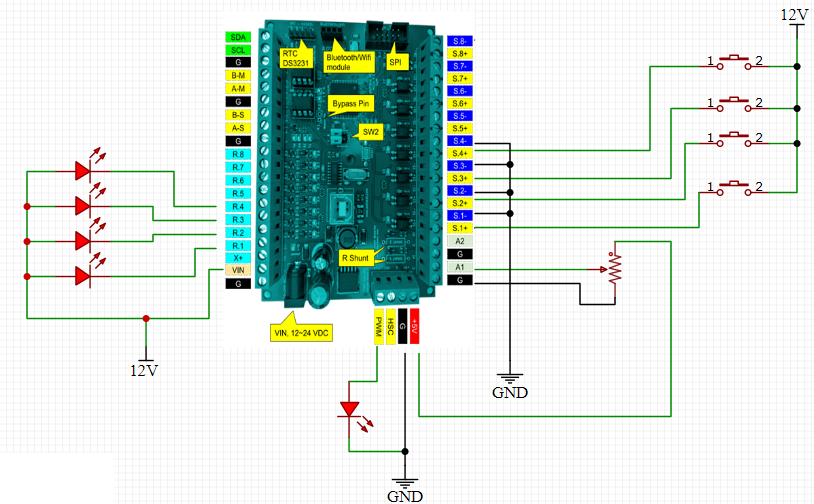
Sebelum :



Sesudah :



## Wiring Hardware



Pada modul pelatihan ini pengkabelan atau wiring sebagai berikut. Adapun penjelasan dari pengkabelan ini adalah. Pengkabelan dibagi menjadi 4 bagian yaitu : digital read, digital write, analog read, dan analog write. Digital read digunakan untuk membaca posisi tombol, digital write digunakan untuk menyalakan dan mematikan lampu, analog read digunakan untuk membaca nilai potensiometer, dan analog write digunakan untuk memberikan nilai pwm pada led

# Program tampilan dasar pada QML

## Membuat Teks

Text{

id : text1

x:100

y:200

text:"Hello World"

color: "#00FF00"

font.family : "Comic Sans MS"

font.pixelSize: 35

font.bold : true

}

Adapun penjelasan dari tiap komponen dasar pada tabel dibawah

Properties dalam membuat teks.

|  |  |  |
| --- | --- | --- |
| **Properties** | **Fungsi** | **Contoh pengisian** |
| Id | Sebagai kode unik komponen qml | root, window, menu |
| X | Untuk mengatur posisi horizontal | 100, 200, 300….dst |
| Y | Untuk mengatur posisi vertical | 100,200,300… dst |
| Text | Untuk menampilkan text | “hello world”, “nama saya husni”, atau apapun kalimatnya diiringi dengan petik dua |
| Color | Untuk menampilkan warna dasar pada tulisan | “red”, “green”, “blue”, atau dapat juga menggunakan kode warna heksadesimal seperti "#68F3F8” |
| font.family | Jenis font yang dibutuhkan | “Times New Roman”, “Arial”, “Comic Sans MS”, dan font lainya |
| font.pixelSize | Untuk mengatur besarnya tulisan | Angka nilai besarnya font |
| font.bold | Untuk mengatur tebalnya tulisan | “true” “false” |

## Membuat Button

Dalam sebuah GUI, tombol digunakan untuk mengaktifkan / menonaktifkan sebuah fungsi. Sebagai contoh, saat kita klik tombol, maka akan menuju ke menu selanjutnya. Atau dapat juga saat kita menekan tombol, maka akan menyalakan lampu PLC Outseal. Pada QML, tombol atau button terdiri dari 6 *properties* dasar yaitu id, x, y, text, pallete, dan onClicked. Ini adalah program dasar untuk menampilkan tombol pada QML.

Button {

id: button1

x :100

y :200

text: "button1"

checkable : true

onClicked:{

}

}

Properties dasar membuat tombol / button.

|  |  |  |
| --- | --- | --- |
| Properties | Fungsi | Contoh pengisian |
| Id | Sebagai kode unik komponen qml | root, window, menu |
| X | Untuk mengatur posisi horizontal | 100, 200, 300….dst |
| Y | Untuk mengatur posisi vertical | 100,200,300… dst |
| Text | Untuk menampilkan text | “hello world”, “nama saya husni”, atau apapun kalimatnya diiringi dengan petik dua |
| checkable | Untuk mengatur jenis button | . True untuk toggle switch, false untuk push button |
| OnClicked | Untuk menaruh program yang berjalan ketika tombol di klik | \*penjelasan detail ada pada integrasi python dan qml” |

## Membuat Rectangle

Rectangle adalah sebuah tampilan berupa kotak. Dapat digunakan untuk mewarnai komponen seperti button. Selain untuk mewarnai button, rectangle juga dapat digunakan untuk grouping komponen supaya dapat dipindahkan secara bersamaan. Ini adalah program dasar untuk menampilkan rectangle

Rectangle{

id : rec1

x : 10

y: 10

width : 300

height : 300

color : “red”

border.width : 0

border.color : “transparent”

}

Properties dasar membuat rectangle

|  |  |  |
| --- | --- | --- |
| Properties | Fungsi | Contoh pengisian |
| Id | Sebagai kode unik komponen qml | root, window, menu |
| X | Untuk mengatur posisi horizontal | 100, 200, 300….dst |
| Y | Untuk mengatur posisi vertical | 100,200,300… dst |
| width | Untuk mengatur lebar | 100, 200, dst |
| height | Untuk mengatur tinggi | 100, 200, dst |
| color | Untuk mengatur warna | “red”, “blue”, “green” |
| border.width | Untuk mengatur besar border | 1,2,3 |
| border.color | Untuk mengatur warna border | “red”, “blue”, “green” |

## Membuat Slider

Slider berfungsi untuk menampilkan sebuah besaran / angka yang nantinya akan dikomunikasikan datanya dengan perangkat lain seperti PLC Outseal. Terdiri dari 11 *properties* dasar untuk membuat masukkan slider yaitu id, x, y, width, height, value, from, to, stepSize, orientation, dan onValueChanged. Ini adalah program dasar untuk menampilkan slider pada QML.

Slider {

id: slider1

x:0

y:150

height: 20

width: 300

value: 0

from:10

to: 255

stepSize: 5

orientation: Qt.Horizontal

onValueChanged: {

}

}

Penjelasan properties slider sebagai berikut:

Properties dasar slider.

|  |  |  |
| --- | --- | --- |
| **Properties** | **Fungsi** | **Contoh pengisian** |
| Id | Sebagai kode unik komponen qml | root, window, menu |
| X | Untuk mengatur posisi horizontal | 100, 200, 300….dst |
| Y | Untuk mengatur posisi vertical | 100,200,300… dst |
| width | Untuk mengatur lebar windows dalam satuan pixel | 100,200,300… dst |
| height | Untuk mengatur tinggi windows dalam satuan pixel | 100,200,300… dst |
| value | Untuk mengatur nilai pertama pada slider | 100,200,300… dst |
| From | Untuk mengatur nilai terendah pada slider | 100,200,300… dst |
| To | Untuk mengatur nilai tertinggi pada slider | 100,200,300… dst |
| stepSize | Untuk mengatur nilai langkah pada slider | 100,200,300… dst |
| orientation | Untuk menentukan orientasi slider | Qt.Horizontal  Qt.Vertical |
| onValueChanged | Untuk meletakkan program apabila slider berubah nilai | \*penjelasan detail ada pada integrasi python dan qml |

## Membuat Gauge

Fungsi gauge dalam sebuah GUI sebenarnya mirip dengan slider hanya dengan tampilan yang berbeda. Ada dua jenis gauge yaitu CircularGauge dan Gauge. Terdiri dari 9 *properties* dasar untuk membuat masukkan CircularGauge yaitu id, x, y, width, height, value, minimumValue, maximumValue, dan style. Penjelasan proteprties tersebut seperti berikut:

Properties gauge / circular gauge.

|  |  |  |
| --- | --- | --- |
| Properties | Fungsi | Contoh pengisian |
| Id | Sebagai kode unik komponen qml | root, window, menu |
| X | Untuk mengatur posisi horizontal | 100, 200, 300….dst |
| Y | Untuk mengatur posisi vertical | 100,200,300… dst |
| width | Untuk mengatur lebar windows dalam satuan pixel | 100,200,300… dst |
| height | Untuk mengatur tinggi windows dalam satuan pixel | 100,200,300… dst |
| value | Untuk mengatur nilai pertama pada Gauge | 100,200,300… dst |
| minimumValue | Untuk mengatur nilai paling kecil pada Gauge | 100,200,300… dst |
| maximumValue | Untuk mengatur nilai paling besar pada Gauge | 100,200,300… dst |
| Style | Untuk mengatur step pada Gauge | CircularGaugeStyle { labelStepSize: 10  } |

# Tabel konversi outseal PLC Code ke alamat MODBUS yang digunakan

|  |  |
| --- | --- |
| **OUTSEAL LADDER CODE** | **MODBUS COIL ADDRESS** |
| R9 | 8 |
| R10 | 9 |
| R11 | 10 |
| R12 | 11 |
| B1 | 128 |
| B2 | 129 |
| B3 | 130 |
| B4 | 131 |

|  |  |
| --- | --- |
| **OUTSEAL LADDER CODE** | **MODBUS HOLDING REGISTER** |
| I.1 | 0 |
| I.2 | 1 |

# Jobsheet 0 : Base Code

Target : Membuat Program minimal untuk menjalankan python, qml, dan modbus

## 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

Timer{

id:tmgauge

interval: 200

repeat: true

running: true

onTriggered: {

}

}

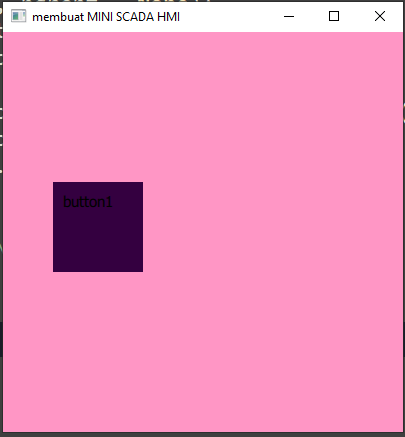
}

# Jobsheet 1 : Digital Output

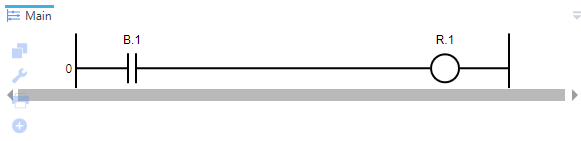
Target : Membuat Program minimal untuk mengendalikan lampu dari python, qml, dan modbus

## Single Output

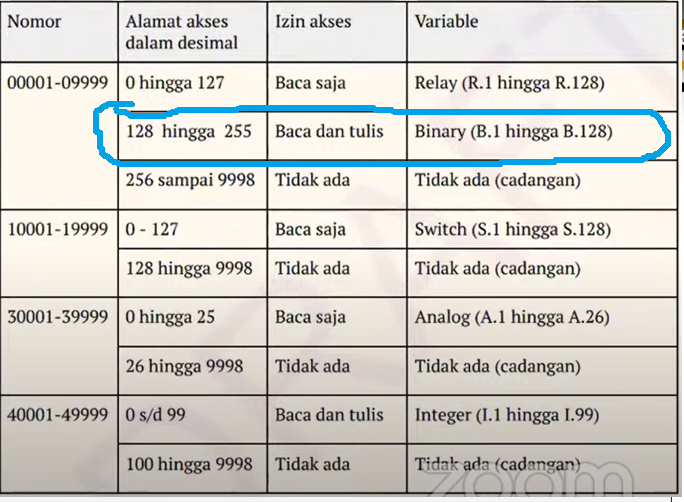
Hasil akhir :



### Program Ladder



B1 mempunyai alamat modbus 128 (coil). jadi B1 mengendalikan R1 yang merupakan pin output 1



### 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")

}

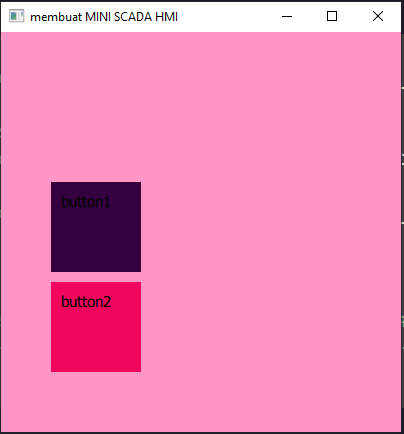
}

}

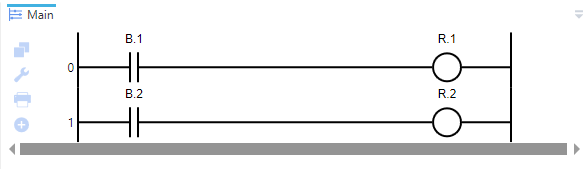
}

## Multiple Output

Hasil Akhir :



### 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")

}

}

}

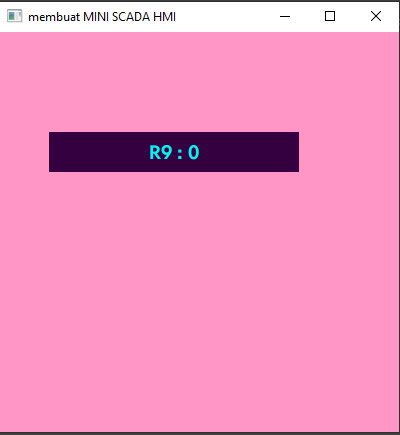
}

# Jobsheet 2 : Digital Input

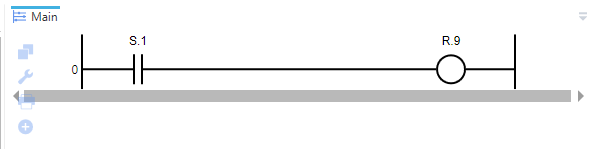
Target : Membuat Program minimal untuk membaca tombol dengan python, qml, dan modbus

## Single Input

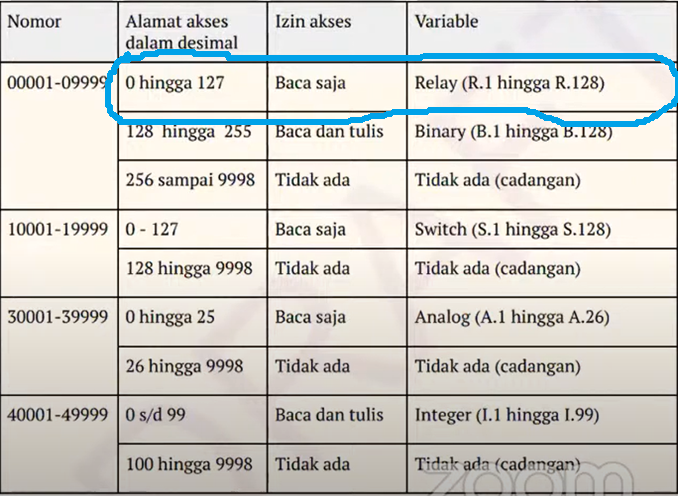
Hasil Akhir :



### Program Ladder



R9 mempunyai alamat modbus 8 (coil). jadi S1 mengendalikan R9 lalu datanya dibaca pada alamat 8



### 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"

}

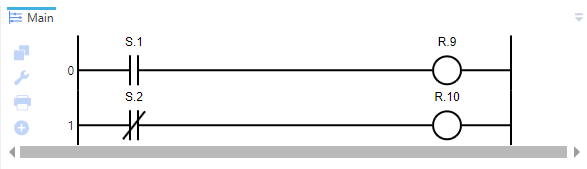
}

}

}

## 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"

}

}

}

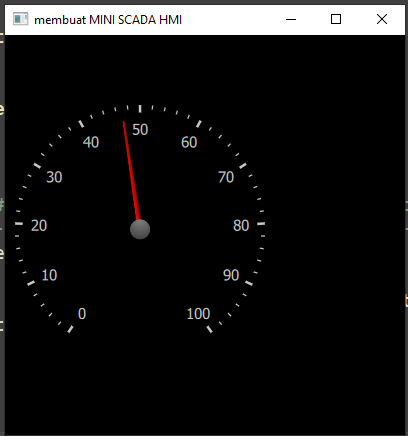
}

# Jobsheet 3 : Analog Input

Target : Membuat Program minimal untuk membaca potensiometer dengan python, qml, dan modbus

## Basic

Hasil Akhir :



### Program Ladder



I.1 mempunyai alamat modbus 0 (holding register). Jadi data potensiometer dimasukkan ke I.1



### 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()

}

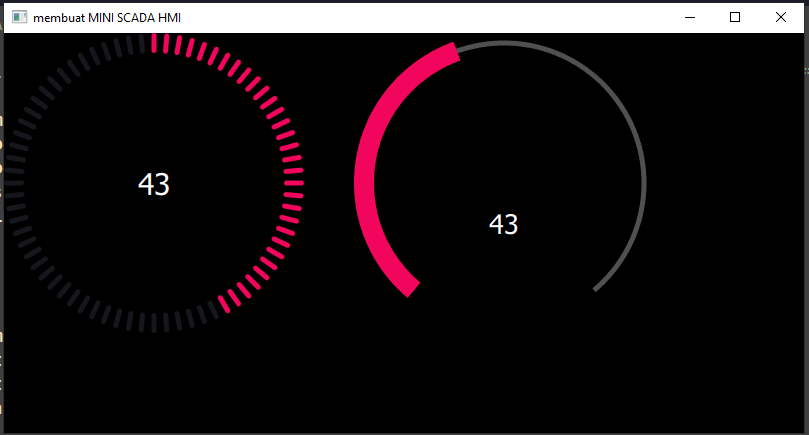
}

}

## Customize

Catatan : masukkan folder “controls” terlebih dahulu agar circularSlider di qml terbaca

Hasil Akhir :



### 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()

}

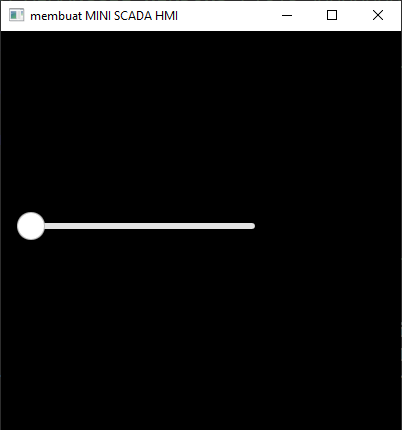
}

# Jobsheet 4 : Analog Output

Target : Membuat Program minimal untuk mengendalikan led pwm dengan python, qml, dan modbus

## Basic

Hasil Akhir :



### 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)

}

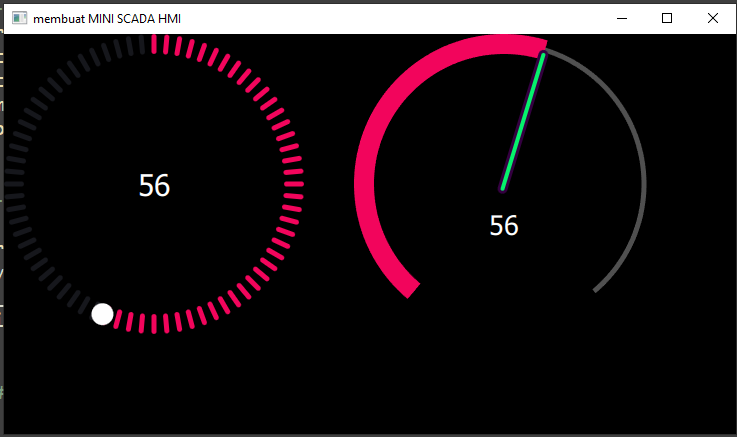
}

}

## Customize

Catatan : masukkan folder “controls” terlebih dahulu agar circularSlider di qml terbaca

Hasil Akhir :



### 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()

}

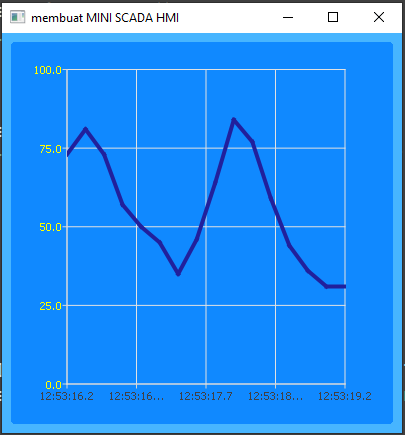
}

}

# Jobsheet 5 : Historical Graph

Target : Membuat Program minimal untuk membaca potensiometer lalu menampilkan ke grafik dengan python, qml, dan modbus

Hasil Akhir :



## 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

}

}

# Tentang Penulis



Muhammad Husni Muttaqin lahir di Bandung, 5 Juni 1998. Husni tertarik dalam dunia elektronika dan otomasi sejak SMP. Husni merupakan seorang Python Programmer di salah satu perusahaan swasta di kota Bandung. Selain Bekerja sebagai python programmer, Husni dan teman temanya yaitu Mas Ajang Rahmat,Mas Rizky Dermawan ,Kang Richard, Pak Febrita Wardana, Pak Febriyansyah, dan Mbak Nisa IK sedang merintis platform komunitas belajar dan sharing online yang bernama “Belajar Arduino dan Mekatronika (ARDUMEKA)”. Husni juga memiliki cita cita menjadi dosen elektronika dan sistem kendali di universitas besar suatu hari nanti.

Apabila ada sesuatu yang ingin didiskusikan Bersama Husni terkait elektronika atau python programming atau bahkan kritik dan saran untuk modul ajar ini silahkan email ke [husnimuttaqin777@gmail.com](mailto:husnimuttaqin777@gmail.com) atau Instagram dengan id : husnimuttaqin. Husni akan sangat senang apabila mendapatkan feedback tentang apa yang ditulisnya. Sekian dan terimakasih.