mainWindow.py

```
import sys

from PyQt5.QtCore import pyqtSlot
from PyQt5.QtWidgets import QMainWindow

from .Ui_mainWindow import Ui_MainWindow

from PyQt5 import QtWidgets
from PyQt5 import QtCore, QtGui
from PyQt5.QtGui import QPen,  QPixmap

from PyQt5.QtCore import  Qt
from threading import Event

from newValue import Controller,  ControlArduino     #import from newValue.py program

ICON_RED_LED = "icons/led-red-on.png"
ICON_GREEN_LED = "icons/green-led-on.png"
```

```
class MainWindow(QMainWindow, Ui_MainWindow):

    def __init__(self, parent=None):
        super(MainWindow, self).__init__(parent)
        self.setupUi(self)

                                                #QPen class defines how should draw lines
        pen = Qpen(Qt.red)
        pen.setWidth(3)
        pen.setCapStyle(Qt.RoundCap)
        scene = QtWidgets.QGraphicsScene()
        pen.setCosmetic(True)
        scene.addPixmap(QPixmap('back.png'))      #draw the background image with Inkscape
        self.item = scene.addLine(60, 170, 97, 97, pen)
        pen = QtGui.QPen(QtGui.QColor(QtCore.Qt.gray))
        brush = QtGui.QBrush(pen.color().darker(100))
        scene.addEllipse(87, 87, 20, 20, pen, brush)
        self.item.setTransformOriginPoint(97, 97)
        self.Grafik.setScene(scene)

        self.stop_flag_time =  Event()
        self.stop_flag_RS232 =  Event()

        self.getController = Controller(self.stop_flag_time) # (1) instance of class Controller
        self.getController.start()                  # (2) Start Thread to update the timer on UI
        self.getController.newTime.connect(self.updateTime) # (3) connect time to(4) updateTime

        self.getArduino = ControlArduino(self.stop_flag_RS232) # (6) instance of class Controller
        self.getArduino.newValue.connect(self.updatePoti)# (8) connect time to (9) updatePoti
        self.getArduino.testRS232.connect(self.updateInfoRS232) # (11) connect to -
                                                                 # - (12) updateInfoRS232
        self.getArduino.start()                     # (7) Start Thread to get data from Arduino (RS232)
```

```
    @pyqtSlot()
    def on_btnExit_clicked(self):
        self.stop_flag_time.set()     # (14) Stop Thread timer
        self.stop_flag_RS232.set()  # (15) Stop Thread data from Arduino (RS232)
        sys.exit(0);
```

```
    def updateTime(self,  timeInterval):        #(4) updateTime
        self.lbltime.setText(timeInterval)          #(5) new time value
```

```
    def updatePoti(self, poti,  potiRotation):   #(9) updateTime
        self.lblAnzeige.setText(str(poti))          #(10) new int value (0 - 1023
        self.item.setRotation(potiRotation)         #(10) new angle of the needle
```

```
    def updateInfoRS232(self, rs232):           #(12) updateInfoRS232
        print(rs232)
        if rs232:                                   # (13) if true LED = green (1)
            self.lblStatusLedUNORS232.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
            self.lblRSinfo.setText("Arduino RS232 okay")
        else:                                       # (13) if true LED = red (0)
            self.lblStatusLedUNORS232.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            self.lblRSinfo.setText("Arduino RS232 failed")
            self.lblAnzeige.setText("Error")
            self.stop_flag_RS232.set()
```

https://www.youtube.com/pi4iot

newValue.py

```
from PyQt5.QtCore import pyqtSignal
from PyQt5.QtCore import  QThread
import datetime,  time
import serial
value = 0
```

```
class Controller(QThread, object):          # (1) instance of class Controller

    now = datetime.datetime.now()
    timeInterval="%0.2d:%0.2d:%0.2d" % (now.hour, now.minute, now.second)
    newTime = pyqtSignal(object)
```

```
    def __init__(self,  event):
        QThread.__init__(self)
        self.stopped = event                        # (14) Stop Thread timer
```

```
    def run(self):                              # (2) Thread update timer
        while not self.stopped.wait(1):
            self.inTime1()
```

```
    def inTime1(self):
        global timeInterval
        now = datetime.datetime.now()
        timeInterval="%0.2d:%0.2d:%0.2d" % (now.hour, now.minute, now.second)
        self.newTime.emit(timeInterval)         # (3) emit new time (5) value
```

```
class ControlArduino(QThread):              # (5)Thread get data from Arduino

    newValue = pyqtSignal( object , object)
    testRS232 = pyqtSignal(object)
```

```
    def __init__(self,  event):
        QThread.__init__(self)
        self.stopped = event                # (15) Stop Thread data from Arduino (RS232)
        self.altValue = 0
```

```
    def run(self):                  # (7) Thread get data from Arduino (RS232
        try:                        # try if com port available
            self.serArduino = serial.Serial('COM6', 115200, timeout=0) #Windows PC
            #ui.serArduino = serial.Serial("/dev/ttyACM0",115200,timeout=1)  #Raspberry
            self.noRS232_UNO = 1
            self.testRS232.emit( 1)  # (11) com port available (13) true (1)
        except:
            print ("RS232 for Arduino not found")
            self.noRS232_UNO = 0
            self.testRS232.emit(0)  # (11) com port available (13) false (0)

        while not self.stopped.wait(0.1):
            self.ArduinoLoop()
```

```
    def ArduinoLoop(self):
        global value
        if self.noRS232_UNO:
            self.serArduino.write(b'p')
            time.sleep(0.01)
            wert = self.serArduino.read(5)
            try:
                wert1 = wert.split()
                intwert = int(wert1[0])
                value = int(22 + (intwert/3.84))        # calculate the new angle
                self.newValue.emit( intwert ,  value) # (8) emit new time (10) value

                print(intwert)
            except:
                print("error Arduino Serial")
```