```python
# -*- coding: utf-8 -*-
"""
Created on Fri Apr  2 21:49:11 2021

@author: JFC-DELL-LATITUDE
"""

import sys

from PyQt5.QtCore import Qt, QTimer
from PyQt5.QtWidgets import (
    QApplication,
    QMainWindow,
    QVBoxLayout,
    QGridLayout,
    QLabel,
    QWidget,
    QSpinBox,
    QComboBox,
    QPushButton,
    QMessageBox,
    QFileDialog
    )

import matplotlib
matplotlib.use('Qt5Agg')
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
from matplotlib.figure import Figure
import serial
import numpy as np
import glob
import sys



class MplCanvas(FigureCanvasQTAgg):
    def __init__(self, parent=None, width=5, height=4, dpi=100):
        fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = fig.add_subplot(111)
        self.axes.set_xlabel("Time (s)")
        self.axes.set_ylabel("Voltage (V)")
        super(MplCanvas, self).__init__(fig)



class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("SPM PyQt5 and Arduino")
        self.canvas = MplCanvas(self, width=5,
                                height=4, dpi=100)
        main_layout = QVBoxLayout()
        main_layout.addWidget(self.canvas)
        control_layout = QGridLayout()
        lbl_com_port = QLabel("COM port:")
        lbl_sample_size =QLabel("Sample size:")
        self.com_port = ""
        self.cb_port = QComboBox()
        self.cb_port.addItems(self.serial_ports())
        self.cb_port.activated.connect(self.add_port)
        self.samples = 1
        spb_samples = QSpinBox()
        spb_samples.setMinimum(1)
        spb_samples.setMaximum(1000)
        spb_samples.setSingleStep(1)
        spb_samples.valueChanged.connect(self.spb_samples_changed)
        self.btn_start = QPushButton("Start")
        self.btn_start.clicked.connect(self.start_acquisition)
        self.btn_stop = QPushButton("Stop")
        self.btn_stop.clicked.connect(self.stop_acquisition)
        self.btn_save = QPushButton("Save")
        self.btn_save.clicked.connect(self.save_file)

        control_layout.addWidget(lbl_com_port,0,0)
        control_layout.addWidget(lbl_sample_size,0,1)
        control_layout.addWidget(self.cb_port,1,0)
        control_layout.addWidget(spb_samples,1,1)
        control_layout.addWidget(self.btn_start,1,2)
        control_layout.addWidget(self.btn_stop,1,3)
        control_layout.addWidget(self.btn_save,1,4)

        main_layout.addLayout(control_layout)
        widget = QWidget()
        widget.setLayout(main_layout)
        self.setCentralWidget(widget)
        self.btn_start.show()
        self.btn_stop.hide()
        self.btn_save.hide()
        self.count = 0
        self.micro_board = None
        self.high_value_board = 5.0
        self.board_resolution = 1024


    def add_port(self):
        self.com_port= str(self.cb_port.currentText())
        print(self.com_port)


    def spb_samples_changed(self,val_samples):
        self.samples = val_samples

    def start_acquisition(self):
        self.stp_acq = False
        try:
            self.micro_board = serial.Serial(str(self.com_port),
                                             9600,timeout=2)
        except:
            dlg_board = QMessageBox()
            dlg_board.setWindowTitle("COM Port Error!")
            str_dlg_board ="The board cannot be read "
            str_dlg_board += "or it wasn't selected!"
            dlg_board.setText(str_dlg_board)
            dlg_board.setStandardButtons(QMessageBox.Ok)
            dlg_board.setIcon(QMessageBox.Warning)
            dlg_board.exec_()
            self.micro_board = None

        if (self.com_port != "" and self.micro_board != None):
            self.btn_start.hide()
            self.btn_stop.show()
            self.btn_save.hide()

            if (self.count == 0):
                self.time_val = 0
                self.values = []
                self.x = np.asarray([])
                self.y = np.asarray([])
                if (self.micro_board != None):
                    self.micro_board.reset_input_buffer()
                self.timer = QTimer()
                self.timer.setInterval(1000)
                self.timer.timeout.connect(self.update_plot)
                self.timer.start()
                print()
                print("Time (s) \t Voltage (V)")

    def stop_acquisition(self):
        self.stp_acq = True


    def update_plot(self):

        try:
            temp = str(self.micro_board.readline().decode('cp437'))
            temp = temp.replace("\r\n","")
            value = (float(temp) *
                          (self.high_value_board/self.board_resolution))
            msg_console = str(self.time_val) + " (s)" + "\t\t "
            msg_console += "{0:0.3f}".format(value) + " (V)"
            print(msg_console)
            self.values.append(str(self.time_val) +","+
                                         str("{0:.3f}".format(value)))
            self.canvas.axes.cla()
            self.x = np.append(self.x,self.time_val)
            self.y = np.append(self.y,value)
            self.canvas.axes.set_xlabel("Time (s)")
            self.canvas.axes.set_ylabel("Voltage (V)")
            self.canvas.axes.plot(self.x,self.y,'C1--o')
            self.canvas.draw()
        except:
            pass
        self.count += 1
        self.time_val += 1
        if (self.count >= self.samples or self.stp_acq == True):
                self.timer.stop()
                self.count = 0
                self.stp_acq = False
                if (self.micro_board != None):
                    self.micro_board.close()
                self.btn_start.show()
                self.btn_stop.hide()
                self.btn_save.show()

    def serial_ports(self) -> list:
        """ Lists serial port names
            :raises EnvironmentError:
            On unsupported or unknown platforms
            :returns:
                A list of the serial ports available on the system
        """
        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

    def save_file(self):
        self.btn_start.hide()
        self.btn_save.hide()
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        fileName, _ = QFileDialog.getSaveFileName(self,
                                    "QFileDialog.getSaveFileName()",""
                        ,"All Files (*);;csv Files (*.csv)", options=options)
        if (fileName):
            file = open(fileName,'w')
            file.write("Time (s),Voltage (V)"+"\n")
            for i in range(len(self.x)-1):
                file.write(str(self.x[i])+","
                               +'{:0.6f}'.format(self.y[i]) +"\n")
            file.write(str(self.x[len(self.x)-1])+","
                           +'{:0.6f}'.format(self.y[len(self.x)-1]))
            file.close()

        self.btn_start.show()
        self.btn_save.show()


    def closeEvent(self, event):
        try:
            if (self.micro_board != None):
                self.micro_board.close()
        except:
            pass

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec_()
```