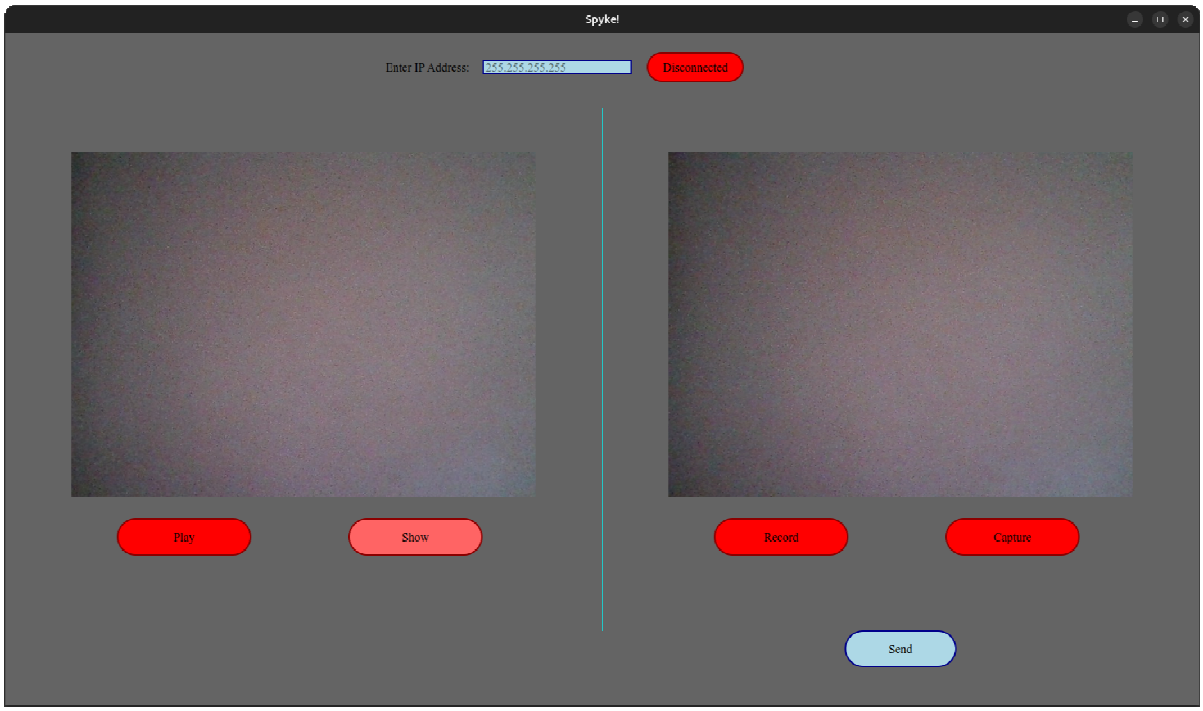
گزارشکار طراحی GUI

امیرمحمد رستم‌آبادی

سیدعلیرضا تهامی

مجتبی دهقانی فیروزآبادی



رابط کاربری به دو قسمت تقسیم شده‌است که سمت راست آن برای ضبط صدا و ذخیره تصویر و فرستادن این دو استفاده میشود و سمت چپ برای نمایش تصویر و پخش صدا استفاده میشود.

در بالای این دو قسمت نیز یک باکس برای وارد کردن IP مقصد و یک کلید برای متصل کردن به IP مقصد وجود دارد.

کلید ها:

1- Disconnected: برای اتصال به مقصد استفاده میشود و پس از فشردن آن IP وارد شده ذخیره شده و این کلید به حالت سبز رنگ تغییر وضعیت میدهد.
2- Capture: پس از کلیک روی آن یک فریم از وبکم را در محلی از پیش مشخص شده ذخیره میکند.
3- Record: پس از کلیک روی آن به مدت سی ثانیه صدا را ضبط میکند.
4- Send: صدا و تصویر ذخیره شده را ارسال میکند.
5- Show: تصویر ذخیره شده در محلی مشخص را نمایش میدهد.
6- Play: صدای دریافت شده را پخش میکند.

Class MainWindow:

پنجره اصلی را در این کلاس ایجاد میکنیم و تمامی object ها را به آن اضافه میکنیم و به کمک آن جایابی object ها و سایز بندی آن ها را انجام میدهیم.

در این کلاس از کلیدهای آماده موجود در PyQt استفاده میکنیم (QPushButton) و بوسیله button.clicked.connect آن‌ها را به یک تابع متصل میکنیم تا در صورت فشرده شدن، آن تابع اجرا شود. بوسیله setStyleSheet. شکل و رنگ کلیدها را مشخص میکنیم. بوسیله setFont نیز فونت و سایز متن‌ها را مشخص مکنیم. به کمک setGeometry نیز محل قرار گیری و سایز object ها را مشخص میکنیم.

بوسیله object از نوع QLabel یک متن را روی صفحه مینویسیم.

بوسیله object از نوع QLineEdit یک باکس برای وارد کردن مقدار IP ایجاد میکنیم.

به کمک object از جنس webcamWidget یک وبکم ایجاد میکنیم و بوسیله setParent. آن را به پنجره اصلی میچسبانیم.

Object های از جنس FillButton نیز نوعی کلید هستند که پس از فشرده شدن تا چند ثانیه بعد به صورت تدریجی پر میشوند که این‌ها برای ضبط و پخش صدا کاربرد دارند. پس از فشردن این کلیدها تا چند ثانیه این‌ها قابل فشردن مجدد نیستند زیرا میخواهیم که صدا بصورت کامل ضبط و یا پخش شود.

در تابع paintEvent نیز بوسیله QPainter یک خط در بین دو قسمت ارسال و دریافت میکشیم.

```python
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from WebcamWidget import WebcamWidget
from FillButton import FillButton
from ImageViewer import ImageViewer


2 usages  AmirMohammadRostamabadi *
class MainWindow(QMainWindow):
    AmirMohammadRostamabadi *
    def __init__(self):
        super().__init__()
        self.ip_value = "255.255.255.255"

        font = self.font()
        font = QFont("Times New Roman", 12)

        # Set window title and size
        self.setWindowTitle("Spyke!")
        self.setGeometry(200, 100, 1600, 900)
        self.setStyleSheet("background-color: rgb(100,100,100)")

        # Create webcam widget
        self.webcam_widget = WebcamWidget()
        self.webcam_widget.setParent(self)  # Adds it to the
        self.webcam_widget.setGeometry(880, 150, self.webcam_widget.width(), self.webcam_widget.height())  # 640 * 480
```

MainWindow › capture_button_clicked()

```
        # Create a button for showing the image
        self.show_button = QPushButton('Show', self)
        self.show_button.setGeometry(460, 650, 180, 50)  # Set button position and size
        self.show_button.clicked.connect(self.show_button_clicked)  # Connect button click event to a function
        self.show_button.setStyleSheet("""
            QPushButton {
                background-color: rgb(255,0,0);
                border: 2px solid darkred;
                border-radius: 25px;
            }
            QPushButton:hover {
                background-color: rgb(255,100,100);
            }
        """)
        self.show_button.setFont(font)

        # Create a button for playing
        self.play_button = FillButton("Play", self)
        self.play_button.setGeometry(150, 650, 180, 50)
        self.play_button.setFont(font)
        self.play_button.clicked.connect(self.play_button_clicked)


    AmirMohammadRostamabadi
    def paintEvent(self, event):
        painter = QPainter(self)
        painter.setPen(QPen(Qt.cyan, 1, Qt.SolidLine))  # Pen color, width, and style
```

MainWindow › capture_button_clicked()

Class WebcamWidget:

در این کلاس بوسیله‌ی کتابخانه‌ی openCV وبکم را فراخوانی میکنیم. در اینجا ابتدا وبکم را initialize میکنیم و تنظیمات مورد نیاز برای نمایش آن را انجام میدهیم (ساخت QVBoxLayout و ...) و سپس یک تایمر میسازیم که هر 25 میلی ثانیه را میشمارد و در تابع updateframe یک فریم جدید از وبکم دریافت میکنیم و پس از انجام تنظیمات مورد نیاز آن را روی صفحه نمایش میدهیم.

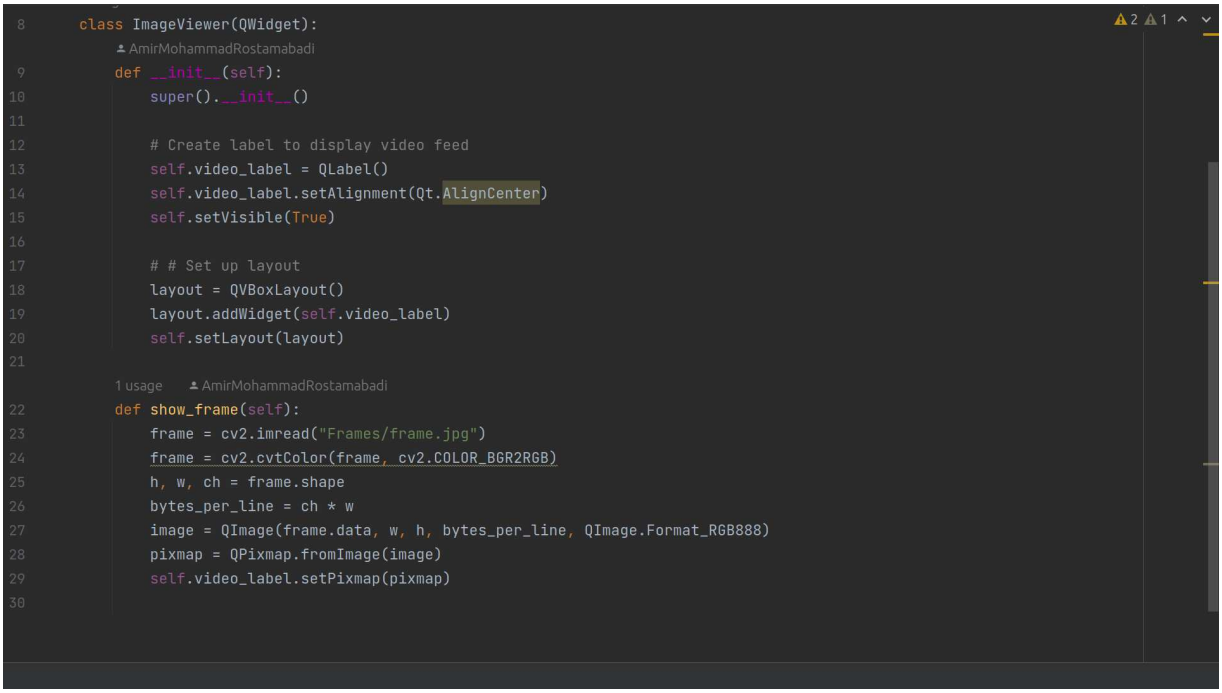
در صورتی که کلید capture فشرده تابع captureframe نیز فراخوانی میشود و یک فریم از وبکم را در محلی از پیش مشخص شده ذخیره میکند.

```
        self.setLayout(layout)

        # Create a timer to update the video feed
        self.timer = QTimer(self)
        self.timer.timeout.connect(self.update_frame)
        self.timer.start(25)  # Update every 25 milliseconds

        # A flag used for capturing the video
        self.capture_flag = False

    1 usage    AmirMohammadRostamabadi
    def update_frame(self):
        ret, frame = self.cap.read()
        if ret:
            if self.capture_flag:
                cv2.imwrite("Frames/frame.jpg", frame)
                self.capture_flag = False
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            h, w, ch = frame.shape
            bytes_per_line = ch * w
            image = QImage(frame.data, w, h, bytes_per_line, QImage.Format_RGB888)
            pixmap = QPixmap.fromImage(image)
            self.video_label.setPixmap(pixmap)

    1 usage    AmirMohammadRostamabadi
    def capture_frame(self):
        self.capture_flag = True
```

WebcamWidget › __init__()

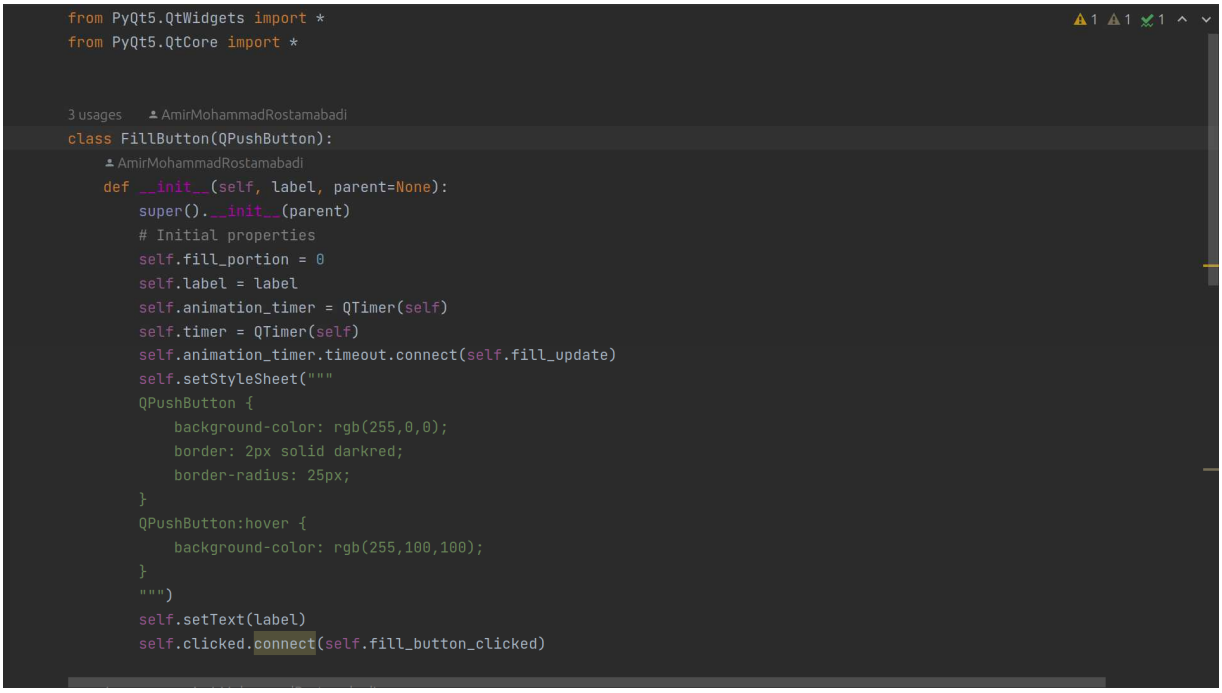
Class ImageViewer:

در این کلاس نیز مانند کلاس WebcamWidget عمل میکنیم و یک باکس برای نمایش تصویر ایجاد میکنیم با این تفاوت که دیگر آن را به وبکم متصل نمیکنیم و پس از هر بار شمردن کلید show یک تصویر را از محلی مشخص میخوانیم و آن را نمایش میدهیم. (به کمک تابع show_frame)



Class FillButton:

در این کلاس نیز نوعی کلید میسازیم که پس از فشرده شدن به صورت یک تایمر را فعال میکند و به صورت آهسته شروع به پر شدن میکند.



```python
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *


3 usages  AmirMohammadRostamabadi
class FillButton(QPushButton):
    AmirMohammadRostamabadi
    def __init__(self, label, parent=None):
        super().__init__(parent)
        # Initial properties
        self.fill_portion = 0
        self.label = label
        self.animation_timer = QTimer(self)
        self.timer = QTimer(self)
        self.animation_timer.timeout.connect(self.fill_update)
        self.setStyleSheet("""
        QPushButton {
            background-color: rgb(255,0,0);
            border: 2px solid darkred;
            border-radius: 25px;
        }
        QPushButton:hover {
            background-color: rgb(255,100,100);
        }
        """)
        self.setText(label)
        self.clicked.connect(self.fill_button_clicked)
```