T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ

PROJE KONUSU: PROGRAMLAMA LAB II

PROJE 3

ÖĞRENCİ ADI: Serhat Arslaner
ÖĞRENCİ NUMARASI: 220502043
GITHUB: github.com/serhatarslaner

ÖĞRENCİ ADI: Emir Dursun
ÖĞRENCİ NUMARASI: 220502001
GITHUB: github.com/emirdrs

DERS SORUMLUSU:
PROF. DR./DR. ÖĞR. ÜYESİ Prof. Dr. Nevcihan DURU

TARİH:02.06.2024

1 GİRİŞ

1.1 Projenin amacı

- Projemizde basit mantık devrelerini tasarlamak için bir platform geliştirdik. Geliştirme sürecinde Python programlama dilini kullandık.
- Projede Mantık Kapıları
- Giriş çıkış elemanları
- Bağlantı elemanları
- Kontrol tuşları bulunmaktadır.

2 GEREKSİNİM ANALİZİ

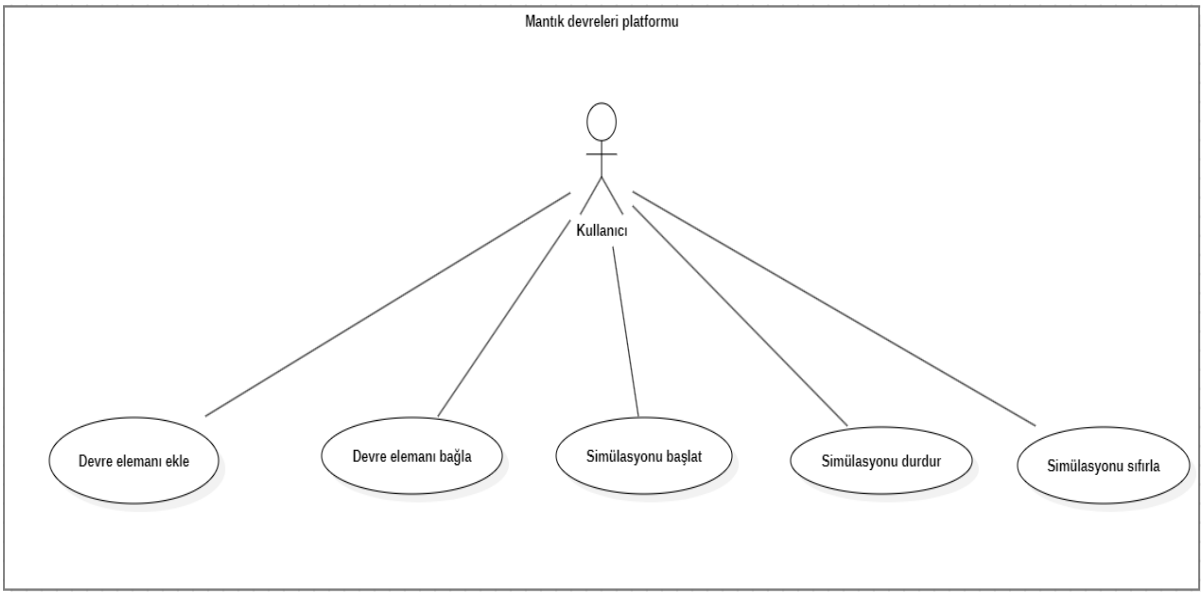
2.1 Arayüz gereksinimleri

- Ana pencere, tüm bileşenlerin yerleştirileceği merkezi bir alan olmalıdır.
- Tüm bileşenler mantıklı ve kullanımı kolay bir şekilde yerleştirilmelidir.
- Kapı seçim radyo düğmeleri, NOT Kapı, BUFFER, AND Kapı…
- Kontrol butonları, Ekle butonu, Bağla butonu….
- Girdi Alanları, Kapı ID, Bağlanacak Kapı ID
- Etiketler, Kapı ID, Bağlanacak Kapı ID,Bağlantılar,Bağlantılar Başlığı
- Grafik Görüntüleme Alanı, QGraphicsView
- Kapı Resimleri
- Dinamik ID Atama
- Çıkış Kutu ve LED Durumu
- Simülasyon İşlevselliği

2.2 Fonksiyonel gereksinimler

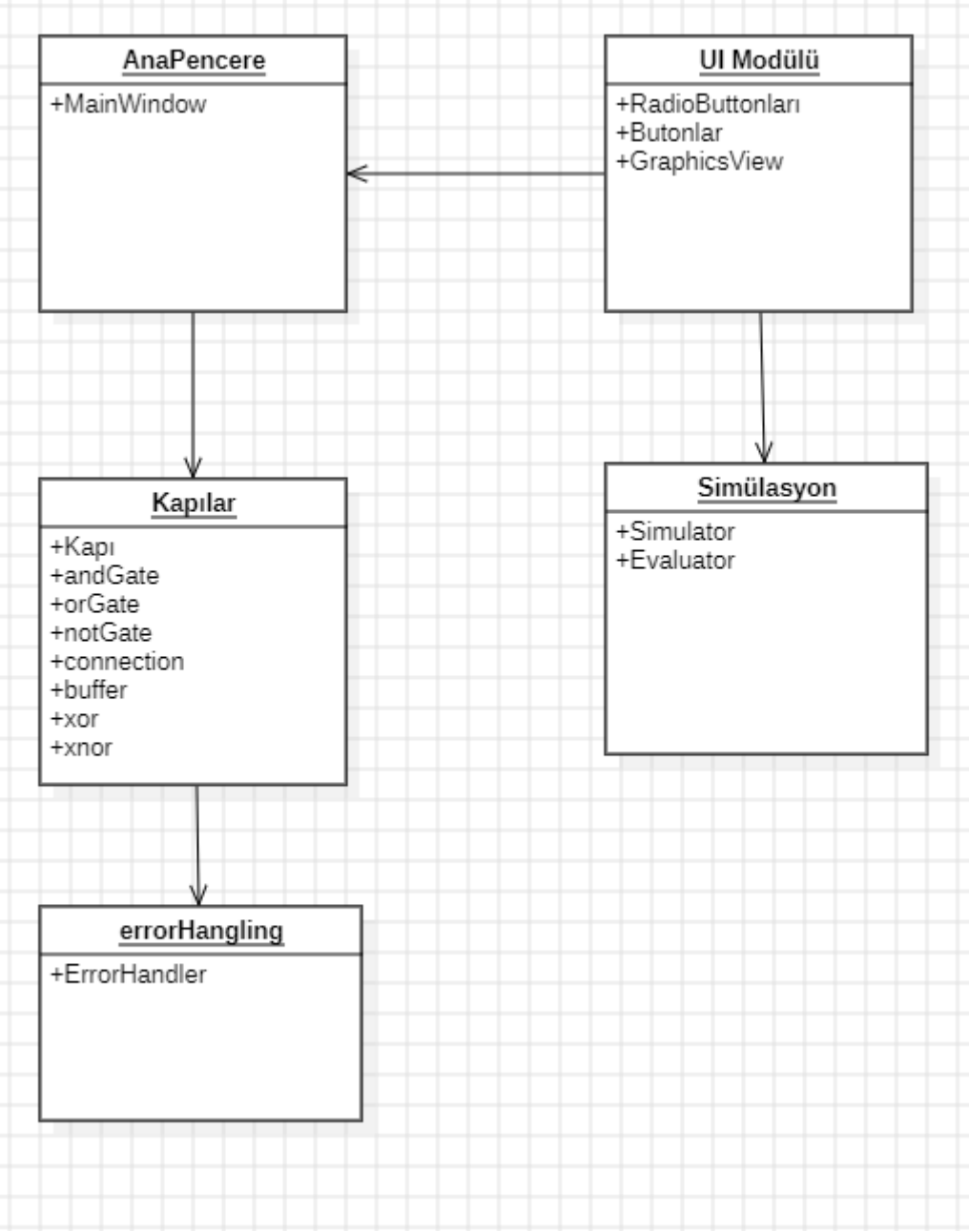
- Kapı Ekleme
- Kapı Bağlama
- Simülasyon Başlatma
- Simülasyonu Durdurma
- Simülasyon Sıfırlama
- Radio Düğmeleri Kapı Seçimi
- Ekle Butonu
- Bağla Butonu
- Başlat Butonu
- Durdur Butonu
- Sıfırla Butonu
- Kapı ID Girdisi
- Bağlanacak Kapı ID Girdisi
- Kapı ve Bağlantı Görüntüleme
- Simülasyon Durumu
- Hata Mesajları
- Dinamik Kapı ve Bağlantı Güncellemeleri
- Kapı ve Bağlantı Kaldırma

2.3 Use-Case diyagramı



3 TASARIM

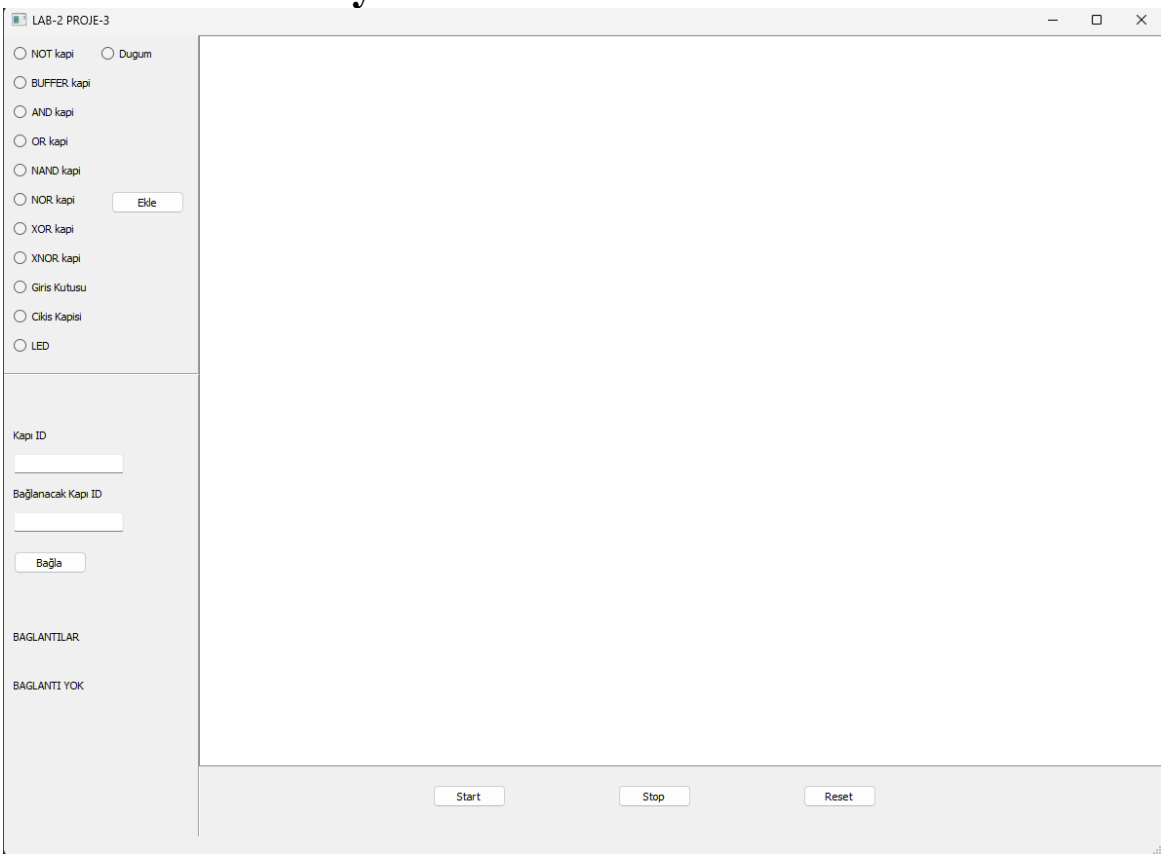
3.1 Mimari tasarım



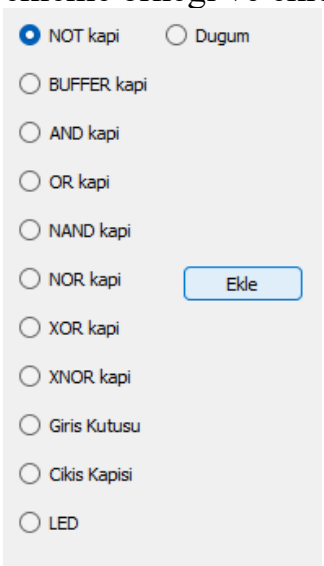
3.2 Kullanılacak teknolojiler

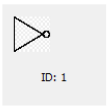
Python programlama dilinde yazılacak.
Python üzerinde arayüz tasarımı ve platformumuzun çalışacağı ekran için PyQT5 kütüphanesi.
PyQT5 ile arayüz tasarımımızı yapmak için QTDesigner.
Visual Studio Code
Uygulamayı kapatma kontrolü için sys kütüphanesi.

3.3 Kullanıcı arayüzü tasarımı

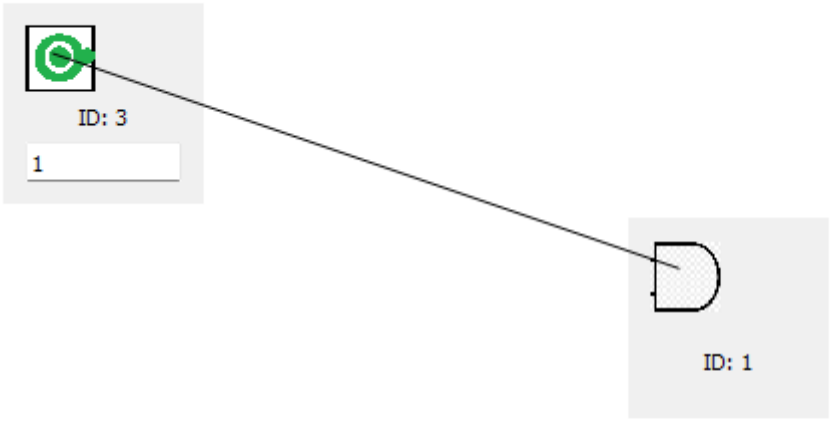


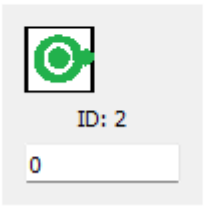
Mantık devreleri platformumuz ilk açıldığında,kullanıcının karşısına devre elemanı ekleyebileceği,devre elemanlarını bağlayabileceği, simülasyonu başlatma/durdurma/sıfırlama işlemlerini yapabileceği ve bağlantıları görüntüleyebileceği sade bir ekran çıkacaktır.Bu ekranda bir devre elemanı ekleme örneği ve ekledikten sonra simülasyon alanında oluşan görüntü:

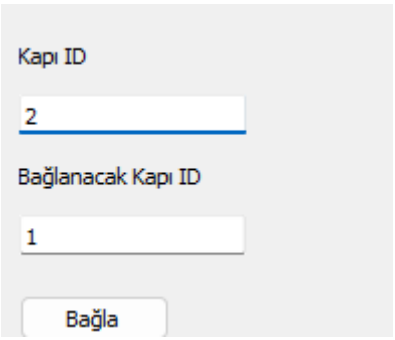




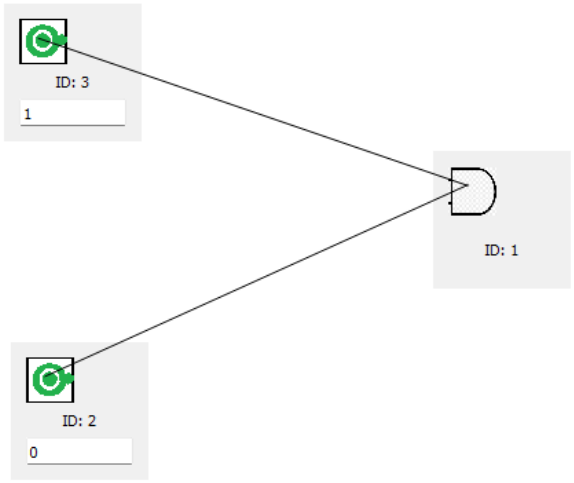
Bağlama işlemine bir örnek olarak bir giriş kutusu ve AND kapısını bağlayalım.
Bağlama işlemi öncesi:







Bağlanacak ID'ler girildikten sonra bağla tuşuna basıldığında oluşan görüntü:

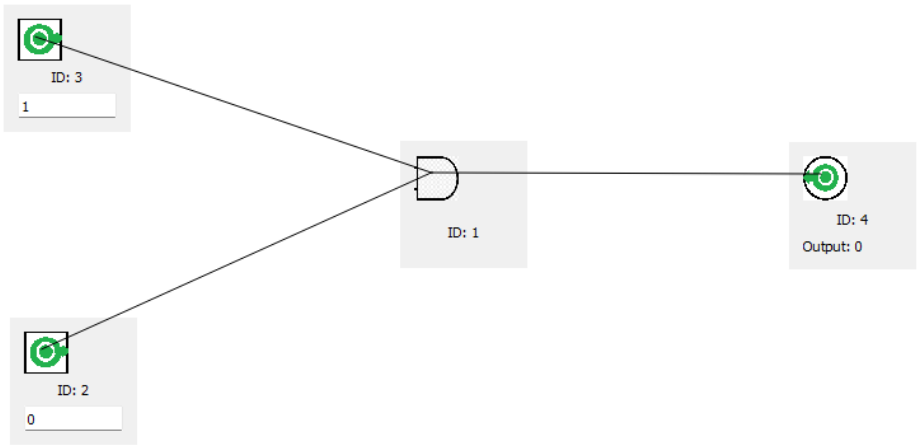


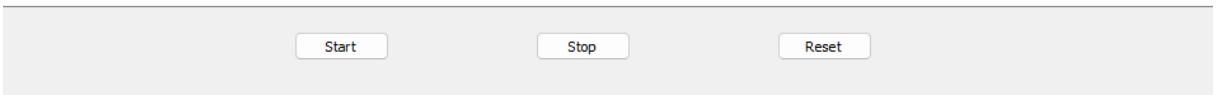
Aynı zamanda simülasyondaki bağlantıların gözüktüğü kısım:

```
BAGLANTILAR


{1: [3, 2]}
```

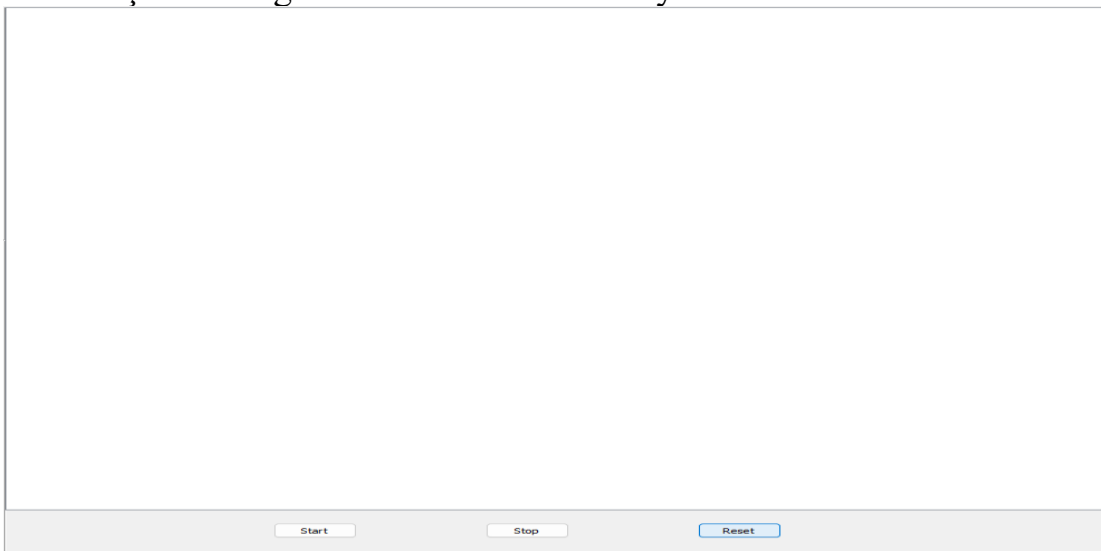
ID 1 e bağlı olan elemanları gösteriyor.

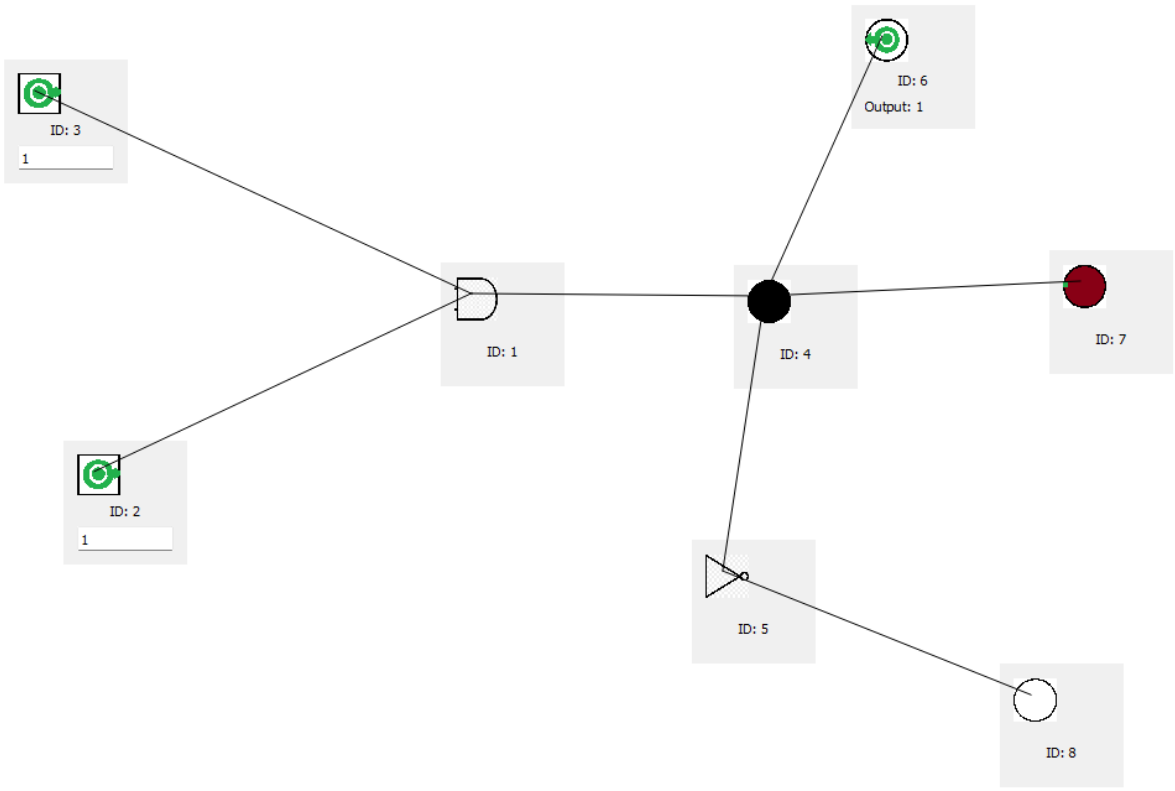




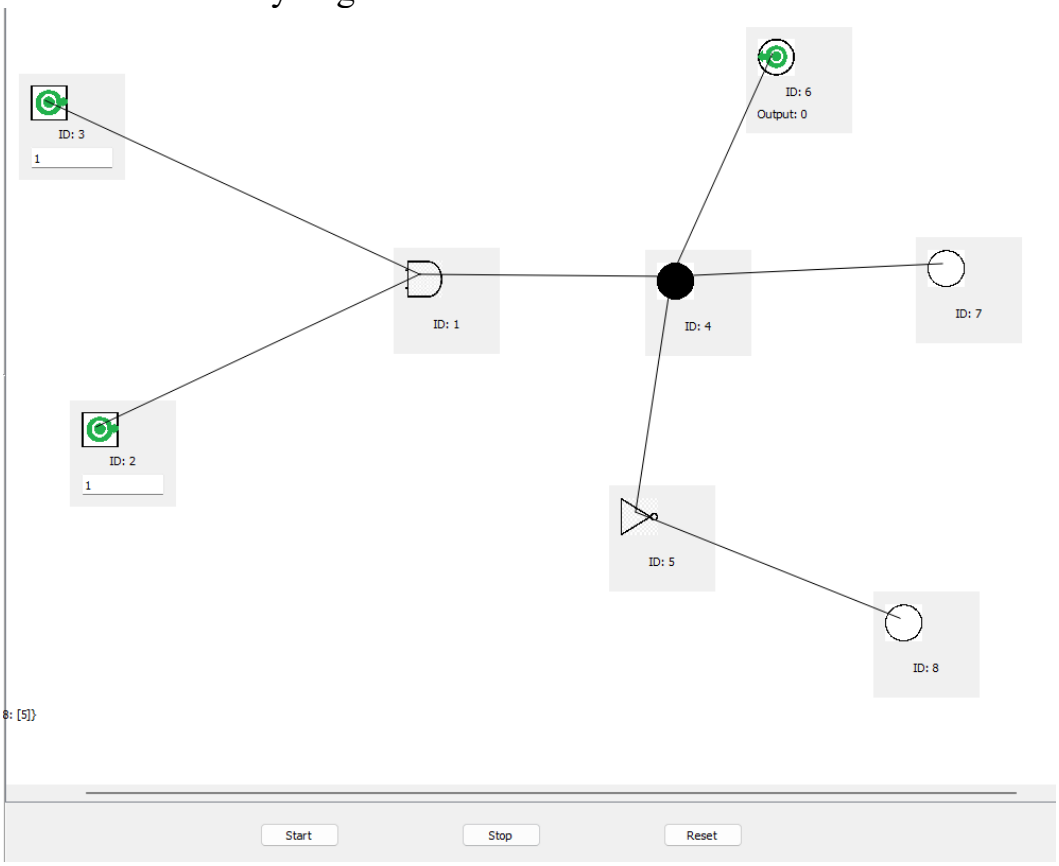
Çıkış kapısını bağladıktan sonra Start tuşuna bastığımızda oluşan görüntü.Çıkış kapısında sonucumuz gözüküyor.

Reset tuşuna bastığımız zaman tüm simülasyon ekranı sıfırlanır.





Örnek bir simülasyon gösterimi.



Stop tuşuna bastığımızda oluşan görüntü.(Devreye giden güç kesilir.)

4 UYGULAMA

4.1 Kodlanan bileşenlerin açıklamaları

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtGui import QPixmap, QPen, QColor
from PyQt5.QtWidgets import QLabel, QVBoxLayout, QWidget, QGraphicsScene, QGraphicsView, QGraphicsLineItem, QLineEdit
from PyQt5.QtCore import pyqtSignal, Qt, QPointF
import sys
```

Kütüphaneler

```
class pictures(QWidget):
    clicked = pyqtSignal()

    def __init__(self, dosyaYolu, x, y, width, height, parent=None):
        super().__init__(parent)
        self.id = None
        self.resim = QPixmap(dosyaYolu).scaled(35, 35)

        self.etiket = QLabel(self)
        self.etiket.setPixmap(self.resim)
        self.etiket.setGeometry(x, y, width, height)

        self.idLabel = QLabel(self)
        self.idLabel.setGeometry(x, y + 70, width, 70)
        self.idLabel.setAlignment(Qt.AlignCenter)

        self.kapsamaAlan = QVBoxLayout(self)
        self.kapsamaAlan.addWidget(self.etiket)
        self.kapsamaAlan.addWidget(self.idLabel)

        self.setFixedSize(100, 100)
        self.sabitlikDurum = True

    def referans(self):
        return self

    def setID(self, id):
        self.id = id
        self.idLabel.setText(f"ID: {self.id}")

    def mousePressEvent(self, event):
        if event.buttons() == Qt.LeftButton:
            self.offset = event.pos()

    def mouseMoveEvent(self, event):
        if event.buttons() == Qt.LeftButton and self.sabitlikDurum:
            widget_pos = self.mapToParent(event.pos() - self.offset)
            self.move(widget_pos)

    def mouseDoubleClickEvent(self, event):
        if self.sabitlikDurum:
            self.sabitlikDurum=False
        else:
            self.sabitlikDurum=True
```

Temel Mantıksal Kapı Sınıfı: Bu sınıf, tüm mantıksal kapılar için temel bir sınıf sağlar. PyQt5 QWidget sınıfını genişletir. Özellikler: clicked adlı bir sinyal tanımlanır. resim adlı bir QPixmap nesnesi ile kapının görsel temsilini tutar. etiket adlı QLabel ile resim gösterilir. idLabel ile kapının ID'si gösterilir. kapsamaAlan adlı QVBoxLayout ile etiket ve ID etiketini içerir. setID() yöntemi ile kapının ID'si ayarlanır. mousePressEvent() ve mouseMoveEvent() yöntemleri, kapının sürüklenip bırakılmasını sağlar. mouseDoubleClickEvent() ile kapı sabitlenebilir veya serbest bırakılabilir.

```python
class notGate(pictures):
    def evaluate(self, inputs):
        return int(not inputs[0])
```

```python
class buffer(pictures):
    def evaluate(self, inputs):
        return inputs[0]
```

```python
class nandGate(pictures):
    def evaluate(self, inputs):

        if len(inputs)==inputs.count(True):
            return 0
        else:
            return 1
```

```python
class orGate(pictures):
    def evaluate(self, inputs):

        if 1  in inputs:
            return 1

        else:
            return 0
```

```python
class andGate(pictures):
    def evaluate(self, inputs):

        if len(inputs)==inputs.count(True):
            return 1
        else:
            return 0
```

```python
class xor(pictures):
    def evaluate(self, inputs):

        if inputs.count(1)==1:
            return 1
        else:
            return 0
```

```python
class nor(pictures):
    def evaluate(self, inputs):

        if 1  in inputs:
            return 0

        else:
            return 1
```

```python
class xnor(pictures):
    def evaluate(self, inputs):
        if inputs.count(1)==1:
            return 0
        else:
            return 1
```

```python
class led(pictures):
    def evaluate(self, inputs):
        if inputs[0] == 0:
            self.etiket.setPixmap(QPixmap("resimler/pasifled.png").scaled(35, 35))
        else:
            self.etiket.setPixmap(QPixmap("resimler/aktifled.png").scaled(35, 35))

        return inputs[0]
```

notkapi, buffer, nandkapi, nor, andkapi, xnor, xor, orkapi Sınıfları

Mantıksal Kapılar: Bu sınıflar pictures sınıfını genişletir ve her biri belirli bir mantıksal kapının işlevselliğini içerir (evaluate yöntemi ile). evaluate() yöntemi, giriş değerlerini alır ve kapının mantıksal işlemini gerçekleştirir.

```
class stop(pictures):
    def evaluate(self, inputs):
        if inputs[0] == 1:
            self.etiket.setPixmap(QPixmap("resimler/pasifled.png").scaled(35, 35))
```

```
class girisKutu(pictures):
    def __init__(self, dosyaYolu, x, y, width, height, parent=None):
        super().__init__(dosyaYolu, x, y, width, height, parent)
        self.inputDeger = QLineEdit(self)
        self.inputDeger.setGeometry(0, 90, 55, 20)
        self.kapsamaAlan.addWidget(self.inputDeger)

    def get_value(self):
        try:
            return int(self.inputDeger.text())
        except ValueError:
            return 0
```

```
class cikisKutu(pictures):
    def evaluate(self, inputs):
        return inputs[0]

class dugum(pictures):
    def __init__(self, dosyaYolu, x, y, width, height, parent=None):
        super().__init__(dosyaYolu, x, y, width, height, parent)
        self.inputs = []

    def add_input(self, kapi):
        self.inputs.append(kapi)

    def evaluate(self, inputs):
        if inputs:
            return inputs[0]
        return 0
```

led, stop, cikisKutu, girisKutu, dugum Sınıfları Özel Kapılar ve Düğüm:

led: Giriş değerine göre aktif veya pasif LED resmi gösterir.

stop: Giriş değeri 1 ise pasif LED resmi gösterir.

cikisKutu: Giriş değerini döndürür.

girisKutu: Kullanıcıdan giriş değeri alır (QLineEdit ile).

dugum: Genel bir düğüm sınıfı, evaluate yöntemi ile giriş değerini döndürür.

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1190, 843)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")

        self.notkapiRadio = QtWidgets.QRadioButton(self.centralwidget)
        self.notkapiRadio.setGeometry(QtCore.QRect(10, 10, 82, 17))
        self.notkapiRadio.setObjectName("notkapiRadio")

        self.bufferRadio = QtWidgets.QRadioButton(self.centralwidget)
        self.bufferRadio.setGeometry(QtCore.QRect(10, 40, 82, 17))
        self.bufferRadio.setObjectName("bufferRadio")

        self.andkapiRadio = QtWidgets.QRadioButton(self.centralwidget)
        self.andkapiRadio.setGeometry(QtCore.QRect(10, 70, 82, 17))
        self.andkapiRadio.setObjectName("andkapiRadio")

        self.orkapiRadio = QtWidgets.QRadioButton(self.centralwidget)
        self.orkapiRadio.setGeometry(QtCore.QRect(10, 100, 82, 17))
        self.orkapiRadio.setObjectName("orkapiRadio")

        self.nandkapiRadio = QtWidgets.QRadioButton(self.centralwidget)
        self.nandkapiRadio.setGeometry(QtCore.QRect(10, 130, 82, 17))
        self.nandkapiRadio.setObjectName("nandkapiRadio")

        self.norRadio = QtWidgets.QRadioButton(self.centralwidget)
        self.norRadio.setGeometry(QtCore.QRect(10, 160, 82, 17))
        self.norRadio.setObjectName("norRadio")

        self.xorRadio = QtWidgets.QRadioButton(self.centralwidget)
        self.xorRadio.setGeometry(QtCore.QRect(10, 190, 82, 17))
        self.xorRadio.setObjectName("xorRadio")

        self.xnorRadio = QtWidgets.QRadioButton(self.centralwidget)
        self.xnorRadio.setGeometry(QtCore.QRect(10, 220, 82, 17))
        self.xnorRadio.setObjectName("xnorRadio")

        self.girisKutusuRadio = QtWidgets.QRadioButton(self.centralwidget)
        self.girisKutusuRadio.setGeometry(QtCore.QRect(10, 250, 82, 17))
        self.girisKutusuRadio.setObjectName("girisKutusuRadio")

        self.cikisKapisiRadio = QtWidgets.QRadioButton(self.centralwidget)
        self.cikisKapisiRadio.setGeometry(QtCore.QRect(10, 280, 82, 17))
        self.cikisKapisiRadio.setObjectName("cikisKapisiRadio")
```

```
self.ledRadio = QtWidgets.QRadioButton(self.centralwidget)
self.ledRadio.setGeometry(QtCore.QRect(10, 310, 82, 17))
self.ledRadio.setObjectName("ledRadio")

self.startButton = QtWidgets.QPushButton(self.centralwidget)
self.startButton.setGeometry(QtCore.QRect(440, 770, 75, 23))
self.startButton.setObjectName("startButton")
self.startButton.clicked.connect(self.start_simulation)

self.stopButton = QtWidgets.QPushButton(self.centralwidget)
self.stopButton.setGeometry(QtCore.QRect(630, 770, 75, 23))
self.stopButton.setObjectName("stopButton")
self.stopButton.clicked.connect(self.stopButonu)

self.resetButton = QtWidgets.QPushButton(self.centralwidget)
self.resetButton.setGeometry(QtCore.QRect(820, 770, 75, 23))
self.resetButton.setObjectName("resetButton")
self.resetButton.clicked.connect(self.reset_simulation)

self.ekleButton = QtWidgets.QPushButton(self.centralwidget)
self.ekleButton.setGeometry(QtCore.QRect(110, 160, 75, 23))
self.ekleButton.setObjectName("ekleButton")
self.ekleButton.clicked.connect(self.nesneButton)

self.line = QtWidgets.QFrame(self.centralwidget)
self.line.setGeometry(QtCore.QRect(190, 0, 20, 831))
self.line.setFrameShape(QtWidgets.QFrame.VLine)
self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
self.line.setObjectName("line")

self.line_2 = QtWidgets.QFrame(self.centralwidget)
self.line_2.setGeometry(QtCore.QRect(200, 740, 991, 16))
self.line_2.setFrameShape(QtWidgets.QFrame.HLine)
self.line_2.setFrameShadow(QtWidgets.QFrame.Sunken)
self.line_2.setObjectName("line_2")

self.line_3 = QtWidgets.QFrame(self.centralwidget)
self.line_3.setGeometry(QtCore.QRect(0, 340, 201, 16))
self.line_3.setFrameShape(QtWidgets.QFrame.HLine)
self.line_3.setFrameShadow(QtWidgets.QFrame.Sunken)
self.line_3.setObjectName("line_3")

self.verticalLayoutWidget = QtWidgets.QWidget(self.centralwidget)
self.verticalLayoutWidget.setGeometry(QtCore.QRect(200, 0, 991, 751))
self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")
```

```
self.graphicsView = QGraphicsView(self.verticalLayoutWidget)
self.graphicsView.setGeometry(QtCore.QRect(0, 0, 991, 751))
self.scene = QGraphicsScene()
self.graphicsView.setScene(self.scene)
self.graphicsView.setRenderHint(QtGui.QPainter.Antialiasing)

self.kapiIsmi = QtWidgets.QLineEdit(self.centralwidget)
self.kapiIsmi.setGeometry(QtCore.QRect(10, 430, 113, 20))
self.kapiIsmi.setObjectName("kapiIsmi")

self.baglanacakKapiIsmi = QtWidgets.QLineEdit(self.centralwidget)
self.baglanacakKapiIsmi.setGeometry(QtCore.QRect(10, 490, 113, 20))
self.baglanacakKapiIsmi.setObjectName("baglanacakKapiIsmi")

self.baglaButton = QtWidgets.QPushButton(self.centralwidget)
self.baglaButton.setGeometry(QtCore.QRect(10, 530, 75, 23))
self.baglaButton.setObjectName("baglaButton")
self.baglaButton.clicked.connect(self.cizgiCiz)

self.baglantilar1 = QtWidgets.QLabel(self.centralwidget)
self.baglantilar1.setGeometry(QtCore.QRect(10,600,300,35))
self.baglantilar1.setObjectName("baglanti1")

self.baglantilar = QtWidgets.QLabel(self.centralwidget)
self.baglantilar.setGeometry(QtCore.QRect(10,650,300,35))
self.baglantilar.setObjectName("baglanti")

self.label = QtWidgets.QLabel(self.centralwidget)
self.label.setGeometry(QtCore.QRect(10, 400, 111, 20))
self.label.setObjectName("label")

self.label_2 = QtWidgets.QLabel(self.centralwidget)
self.label_2.setGeometry(QtCore.QRect(10, 460, 111, 20))
self.label_2.setObjectName("label_2")

self.dugumRadio = QtWidgets.QRadioButton(self.centralwidget)
self.dugumRadio.setGeometry(QtCore.QRect(100, 10, 82, 17))
self.dugumRadio.setObjectName("radioButton")

MainWindow.setCentralWidget(self.centralwidget)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)
```

Ui_MainWindow Sınıfı Ana Kullanıcı Arayüzü:
Uygulamanın ana penceresini oluşturur ve tüm widget'ları ve etkileşimleri yönetir.
Özellikler: notkapiRadio, bufferRadio, vb.: Mantıksal kapılar için radyo butonları.
startButton, stopButton, resetButton, ekleButton, baglaButton: İşlemleri başlatmak, durdurmak, sıfırlamak, nesne eklemek ve bağlantı kurmak için butonlar.
kapiIsmi, baglanacakKapiIsmi: Kullanıcıdan kapı ID'lerini almak için giriş kutuları. graphicsView: Kapıların ve bağlantıların yerleştirildiği alan (QGraphicsView ile).
baglantilar,
baglantilar1: Bağlantıları göstermek için etiketler.
allItems: Tüm kapı nesnelerini tutan liste.
baglantilarr: Bağlantıları (ID'ler arasında) tutan sözlük.
objectID: Nesneler için benzersiz ID.

```
def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "LAB-2 PROJE-3"))
    self.notkapiRadio.setText(_translate("MainWindow", "NOT kapi"))
    self.bufferRadio.setText(_translate("MainWindow", "BUFFER kapi"))
    self.andkapiRadio.setText(_translate("MainWindow", "AND kapi"))
    self.orkapiRadio.setText(_translate("MainWindow", "OR kapi"))
    self.nandkapiRadio.setText(_translate("MainWindow", "NAND kapi"))
    self.norRadio.setText(_translate("MainWindow", "NOR kapi"))
    self.xorRadio.setText(_translate("MainWindow", "XOR kapi"))
    self.xnorRadio.setText(_translate("MainWindow", "XNOR kapi"))
    self.girisKutusuRadio.setText(_translate("MainWindow", "Giris Kutusu"))
    self.cikisKapisiRadio.setText(_translate("MainWindow", "Cikis Kapisi"))
    self.ledRadio.setText(_translate("MainWindow", "LED"))
    self.startButton.setText(_translate("MainWindow", "Start"))
    self.stopButton.setText(_translate("MainWindow", "Stop"))
    self.resetButton.setText(_translate("MainWindow", "Reset"))
    self.ekleButton.setText(_translate("MainWindow", "Ekle"))
    self.baglaButton.setText(_translate("MainWindow", "Bağla"))
    self.label.setText(_translate("MainWindow", "Kapı ID"))
    self.label_2.setText(_translate("MainWindow", "Bağlanacak Kapı ID"))
    self.dugumRadio.setText(_translate("MainWindow", "Dugum"))
    self.baglantilar.setText(_translate("MainWindow","BAGLANTI YOK"))
    self.baglantilar1.setText(_translate("MainWindow","BAGLANTILAR"))
```

```
def nesneButton(self):
    if self.notkapiRadio.isChecked():
        item = notGate('resimler/notGate.png', 0, 0, 55, 55)
    elif self.bufferRadio.isChecked():
        item = buffer('resimler/buffer.png', 0, 0, 55, 55)
    elif self.andkapiRadio.isChecked():
        item = andGate('resimler/andGate.png', 0, 0, 55, 55)
    elif self.orkapiRadio.isChecked():
        item = orGate('resimler/orGate.png', 0, 0, 55, 55)
    elif self.nandkapiRadio.isChecked():
        item = nandGate('resimler/nandGate.png', 0, 0, 55, 55)
    elif self.norRadio.isChecked():
        item = nor('resimler/nor.png', 0, 0, 55, 55)
    elif self.xorRadio.isChecked():
        item = xor('resimler/xor.png', 0, 0, 55, 55)
    elif self.xnorRadio.isChecked():
        item = xnor('resimler/xnor.png', 0, 0, 55, 55)
    elif self.girisKutusuRadio.isChecked():
        item = girisKutu('resimler/girisKutu.png', 0, 0, 55, 55)
    elif self.cikisKapisiRadio.isChecked():
        item = cikisKutu('resimler/cikisKutu.png', 0, 0, 55, 55)
        cikisLabel = QLabel(item)
        cikisLabel.setGeometry(0, 110, 55, 20)
        item.kapsamaAlan.addWidget(cikisLabel)
        self.cikisLabels.append(cikisLabel)
    elif self.ledRadio.isChecked():
        item = led('resimler/pasifled.png', 0, 0, 55, 55)

    elif self.dugumRadio.isChecked():
        item = dugum("resimler/dugum.png",0,0,55,55)
```

Yeni Kapı Ekleme: Kullanıcı bir radyo butonuna tıkladığında, seçilen kapıyı ekler ve ID'yi ayarlar.

```
def cizgiCiz(self):
    kapiID = int(self.kapiIsmi.text())
    baglanacakKapiID = int(self.baglanacakKapiIsmi.text())

    kapi = next((item for item in self.allItems if item.id == kapiID), None)
    baglanacak_kapi = next((item for item in self.allItems if item.id == baglanacakKapiID), None)

    if kapi and baglanacak_kapi:
        if baglanacakKapiID not in self.baglantilarr:
            self.baglantilarr[baglanacakKapiID] = []
        self.baglantilarr[baglanacakKapiID].append(kapiID)

        line = QGraphicsLineItem()
        line.setPen(QPen(QColor("black"), 1))
        self.scene.addItem(line)
        line.setLine(kapi.x() + 25, kapi.y() + 25, baglanacak_kapi.x() + 25, baglanacak_kapi.y() + 25)

    self.baglantilar.setText("{}".format(self.baglantilarr))
```

Bağlantı Oluşturma: Kullanıcıdan kapı ID'lerini alır ve bu kapılar arasında bir bağlantı oluşturur (QGraphicsLineItem)

```
def start_simulation(self):
    def evaluate_kapi(kapi):
        inputs = []
        if kapi.id in self.baglantilarr:
            for input_id in self.baglantilarr[kapi.id]:
                inputKapi = next((item for item in self.allItems if item.id == input_id), None)
                if inputKapi:
                    inputs.append(evaluate_kapi(inputKapi))
        if isinstance(kapi, girisKutu):
            return kapi.get_value()
        return kapi.evaluate(inputs)

    for item in self.allItems:
        if isinstance(item, cikisKutu):
            result = evaluate_kapi(item)
            cikisLabel = next((label for label in self.cikisLabels if label.parent() == item), None)
            if cikisLabel:
                cikisLabel.setText(f"Output: {result}")

    for item in self.allItems:
        if isinstance(item, led):
            result = evaluate_kapi(item)
            cikisLabel = next((label for label in self.cikisLabels if label.parent() == item), None)
            if cikisLabel:
                cikisLabel.setText(f"Output: {result}")

    self.baglantilar.setText("{}".format(self.baglantilarr))
```

Simülasyonu Başlatma: Tüm kapıları değerlendirir ve sonuçları çıkış kapıları ve LED'lere yazar.

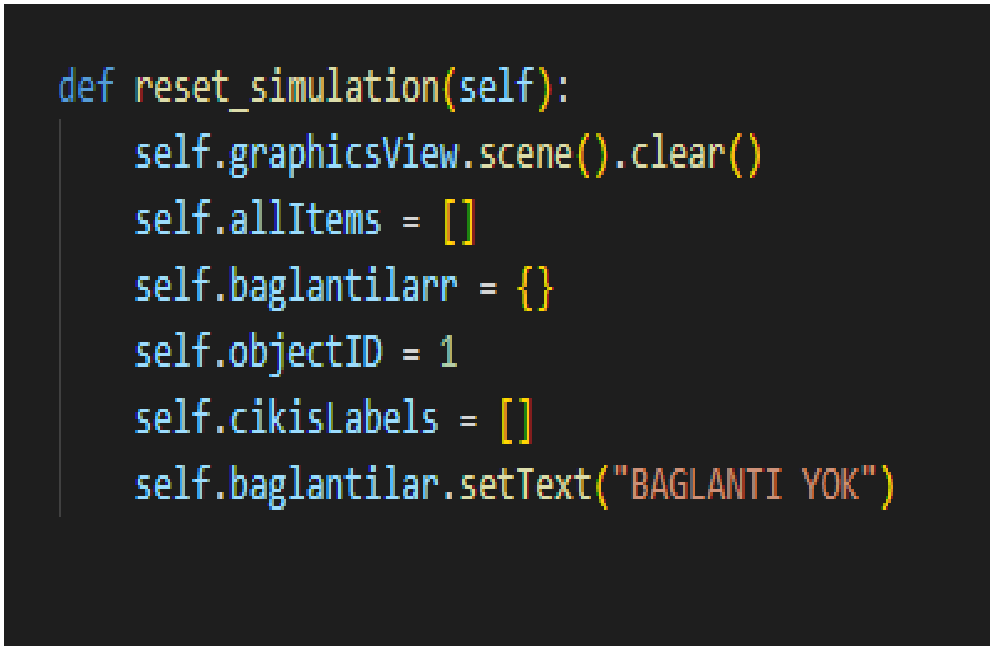
```
def stopButonu(self):
    def evaluate_kapi(kapi):
        inputs = []
        if kapi.id in self.baglantilarr:
            for input_id in self.baglantilarr[kapi.id]:
                inputKapi = next((item for item in self.allItems if item.id == input_id), None)
                if inputKapi:
                    inputs.append(evaluate_kapi(inputKapi))

        if isinstance(kapi, girisKutu):
            return kapi.get_value()
        return kapi.evaluate(inputs)

    for item in self.allItems:
        if isinstance(item,led):
            item.evaluate([0])

        elif isinstance(item, cikisKutu):
            cikisLabel = next((label for label in self.cikisLabels if label.parent() == item), None)
            if cikisLabel:
                cikisLabel.setText(f"Output: {0}")
```

Simülasyonu Durdurma: Tüm çıkış kapılarını ve LED'leri sıfırlar.



Simülasyonu Sıfırlama: Tüm nesneleri ve bağlantıları temizler ve başlangıç durumuna döner.

```python
def main():
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```

Uygulama Giriş Noktası: Uygulama başlatılır ve ana pencere gösterilir.

4.2 Görev dağılımı

- Bileşenlerinin tasarım ve geliştirme aşamalarında Kütüphanelerin öğrenilmesi , Algoritma hazırlanması ve deneme yöntemleri ortak gerçekleştirilmiştir.
- Raporun hazırlanması süreçlerinde ortak çalışma yapılmıştır.

4.3 Karşılaşılan zorluklar ve çözüm yöntemleri

- Ledlerin yanıp sönmesi işlemi – çıkış kapısında kullanılan fonksiyonu değiştirerek çözüme ulaştırıldı.
- Uygulamanın çökmesi – koddaki mantık hataları giderilerek düzeltilmiştir.

4.4 Proje isterlerine göre eksik yönler

- Tüm isterler karşılanmıştır.

5 TEST VE DOĞRULAMA

5.1 Yazılımın test süreci

- Proje kapsamında Yazılan Tüm Sınıflar ve fonksiyonlar test edilmiştir

5.2 Yazılımın doğrulanması

- Proje kapsamında Testi Yapılan Sınıflar ve Fonksiyonlar üzerinde tespit edilen anormallikler giderilmiştir.