Федеральное государственное бюджетное образовательное учреждение
высшего образования Финансовый университет при Правительстве Российской Федерации
Факультет Информационных технологий и анализа больших данных

Контрольная работа
По дисциплине
Технологии разработки приложений для мобильных устройств
“Основы технологий интернета вещей”
Схема поддержки питания электросети.

Выполнил:
Студент 3 курса
Группы ПИ20-6
Савин Алексей

г. Москва
2022

Цель работы:

Создание системы поддержания постоянного питания электросети на базе микроконтроллера Arduino.

В электросети могут происходить перепады напряжения, вплоть до того, что напряжение может вовсе упасть до 0. Для поддержания систем, которые постоянно должны быть включены, создаются резервные источники питания. В данной работе целью является создание симуляции такой электросети, которая покажет наглядно, как работает такая экстренная система.

Задачи:

- Составление электросхемы, содержащей два источника питания – датчика;

- Написание скрипта для платы Arduino Uno;

- Написание модели предсказания перепадов напряжения;

- Запуск и тест симуляции работы схемы.

Архитектура проекта:

Проект состоит из:

- Схемы в Proteus 8;

- Скетч-файла Arduino;

- Скрипта python.

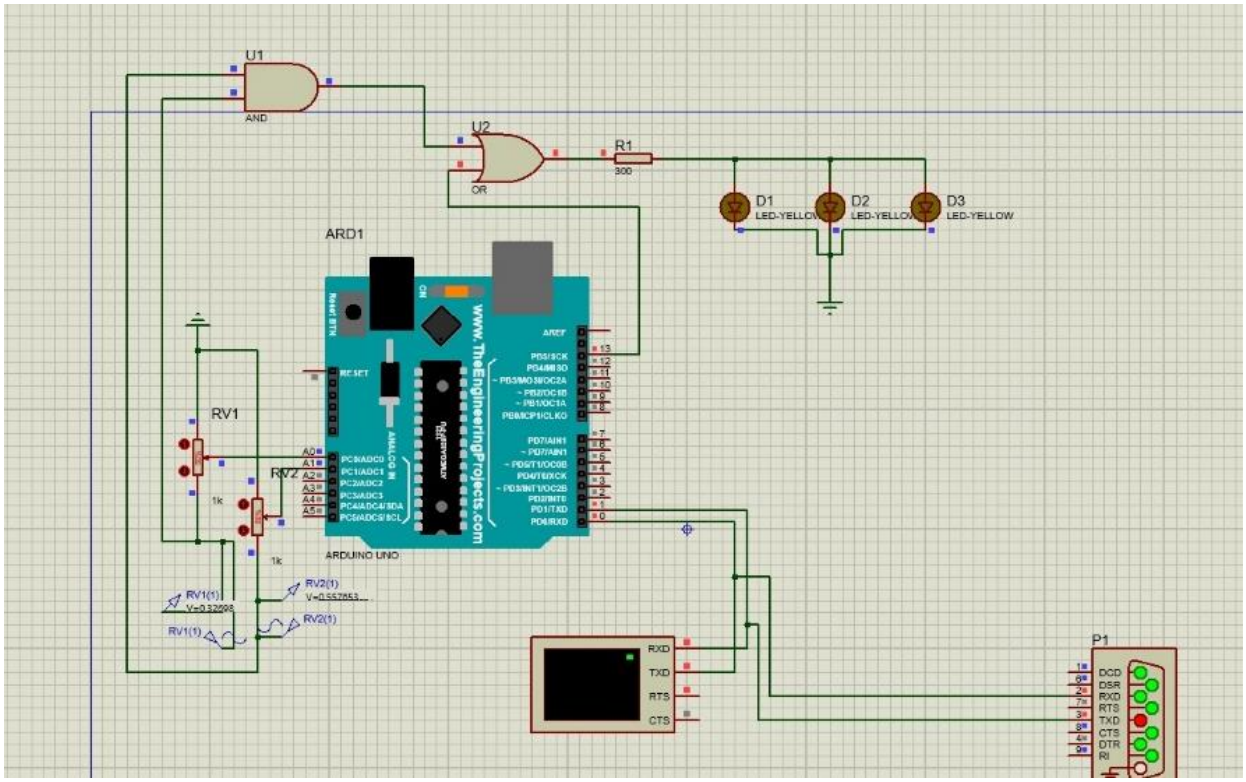


Схема состоит из:

- Двух генераторов напряжения;

- Самой платы;

- Модуля общения с COM-портами COMPIM;

- Системы ламп (для демонстрации работы поддержания постоянного питания).

Генераторы с изначальным напряжением 3 В работают с разной частотой и разной фазой. Они соединены с платой (каждый на разный аналоговый вход), а также с логической схемой AND, от которой сигнал идет в схему OR. Плата соединена со схемой OR на 13 пине, а также 0 и 1 пин соединены с устройством COMPIM и виртуальным терминалом (терминал нужен для начальной отладки внутри эмулятора).

Скетч-файл представляет из себя следующее и реализует функционал:

```
float anlg1;
float anlg2;
unsigned long mils; //инициализация переменных
void setup() {
 Serial.begin(9600); //инициализация вывода сигнала на 13 пин
 pinMode(13, OUTPUT);
}
void loop() {
 anlg1 = analogRead(0); //считывание показаний с первого датчика
 anlg2 = analogRead(1); //считывание показаний со второго датчика
 mils = millis(); //замер времени в момент считывания в миллисекундах
 Serial.print(anlg1); //вывод в серийный порт показаний с датчиков и показателя времени
 Serial.print(";");
 Serial.print(anlg2);
 Serial.print(";");
 Serial.print(mils);
 Serial.println("");
 int timeout = 0; //задание переменной для подсчёта времени ожидания
 char command = 0; //переменная для команды, полученной от терминала
 while (Serial.available()==0) { //цикл ожидания команды, пока команда недоступна увеличивается время тайм-аута
  if (++timeout>10000){
   break;
  }
 }
 if (Serial.available()>0){
  command = Serial.read(); //чтение команды из серийного порта
 }
 timeout = 0; // обнуление тайм-аута
 switch (command) {
  case 1: //проверка команды и последующее действие включить или отключить питание
  digitalWrite(13, HIGH);
```

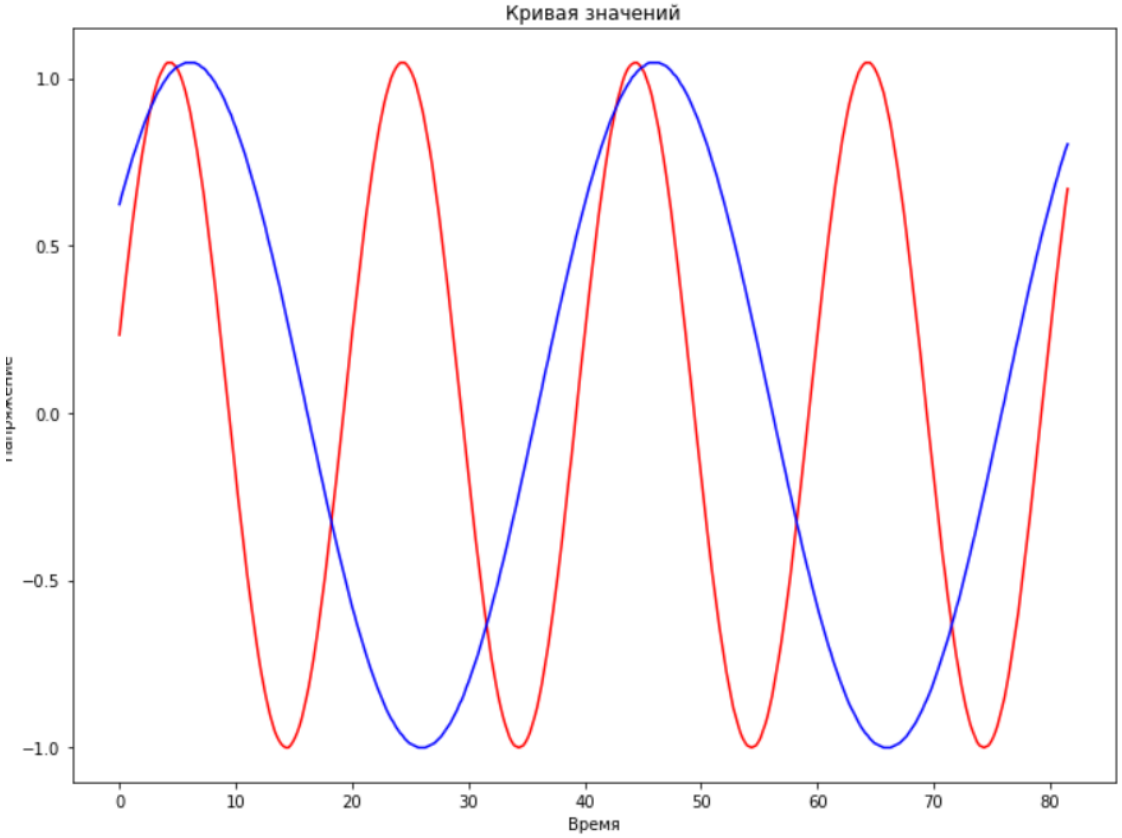
```
   break;
   case 0:
   digitalWrite(13, LOW);
   break;
   default:
   digitalWrite(13, LOW);
   break;
  }
}
```

Плата отправляет свои показания на порт COM1. Параллельно работе симуляции, на порт COM2 подключается скрипт python, прослушивающий порт.

```
import serial
import time
#import numpy as np
import math
#import random
from sklearn.linear_model import LinearRegression

X = []
y = []
for i in range(0, 181):
   X.append([math.sin(math.radians(i * 0.5)), i * 0.5])
   y.append([math.sin(math.radians((i + 1) * 0.5)) * 3 + 3, (i + 1) * 0.5])
reg = LinearRegression().fit(X, y)
#Создание модели линейной регрессии для предсказания показаний с датчиков. Работает
предсказывая следующее показание по синусу.

serialPort = serial.Serial(port="COM2", baudrate=9600, bytesize=8, timeout=2,
stopbits=serial.STOPBITS_ONE)
#Создание объекта-порта
print("Connection established")
serialString = ""  # Переменная, которая будет хранить значения полученные с платы до
дешифровки
crit = 7.0 #критическое значение, ниже которого нужно включать питание
data1 = [] #массив данных для датчика 1, будет содержать до двух пар значений
напряжения:времени
data2 = [] #массив данных для датчика 2, аналогичен первому
comm = 1 команда для отправки на плату, изначально 1
while 1:
   resp = 0
   # Ожидание данных, resp отвечает за то, нужен ли ответ плате
   if serialPort.in_waiting > 0:

      #Считывание данных, если получена новая стока
      serialString = serialPort.readline()
      resp = 1
```

```
      # Вывод содержания полученной строки и добавление её к массивам данных для
предсказания
      try:
         i = serialString.decode("Ascii")
         print(i)
#добавление данных в массивы
         if (len(data1) < 1) or (len(data1) == 1):
            j = i.split(";")
            v1 = ((float(j[0])/100)-3)/3
            v2 = ((float(j[1])/100)-3)/3
            t = float(j[2].rstrip())/100
            data1.append([v1, t])
            data2.append([v2, t])
         elif len(data1) == 2:
            data1[0] = data1[1]
            data2[0] = data2[1]
            j = i.split(";")
            v1 = ((float(j[0])/100)-3)/3
            v2 = ((float(j[1])/100)-3)/3
            t = float(j[2].rstrip()) / 100
            data1[1] = [v1, t]
            data2[1] = [v2, t]
         print(data1)
         print(data2)
#если длина массива 1 равна 2, то происходит запуск предсказания
         if len(data1) == 2:
            pred1 = reg.predict(data1)
            pred2 = reg.predict(data2)
            print(pred1)
            if (pred1[1][0] > crit) & (pred2[1][0] > crit):
               comm = 0
            else:
               comm = 1
         print(comm)
      except: #исключение, если данные не получены или их невозможно расшифровать
         time.sleep(10)
         pass
   else:
#отправка команды плате, если она ждет ответа
      if resp == 1:
         try:
            serialPort.write(comm)
            serialPort.flush()
         except:
            time.sleep(10)
            pass
```

После обработки данных я получил следующий график зависимости напряжения от времени для двух генераторов:



Соответственно, работа схемы в разные моменты времени:

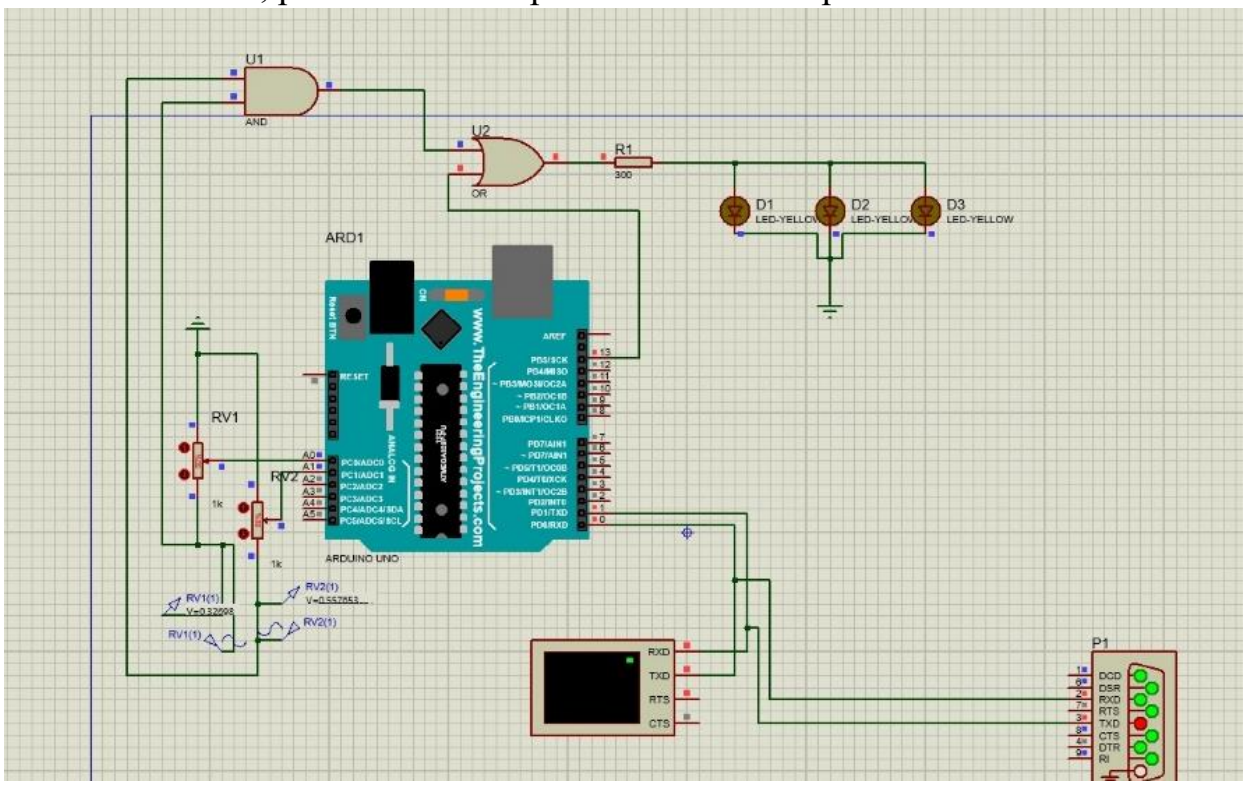


Рисунок 1 Плата подаёт питание, когда на генераторах спад

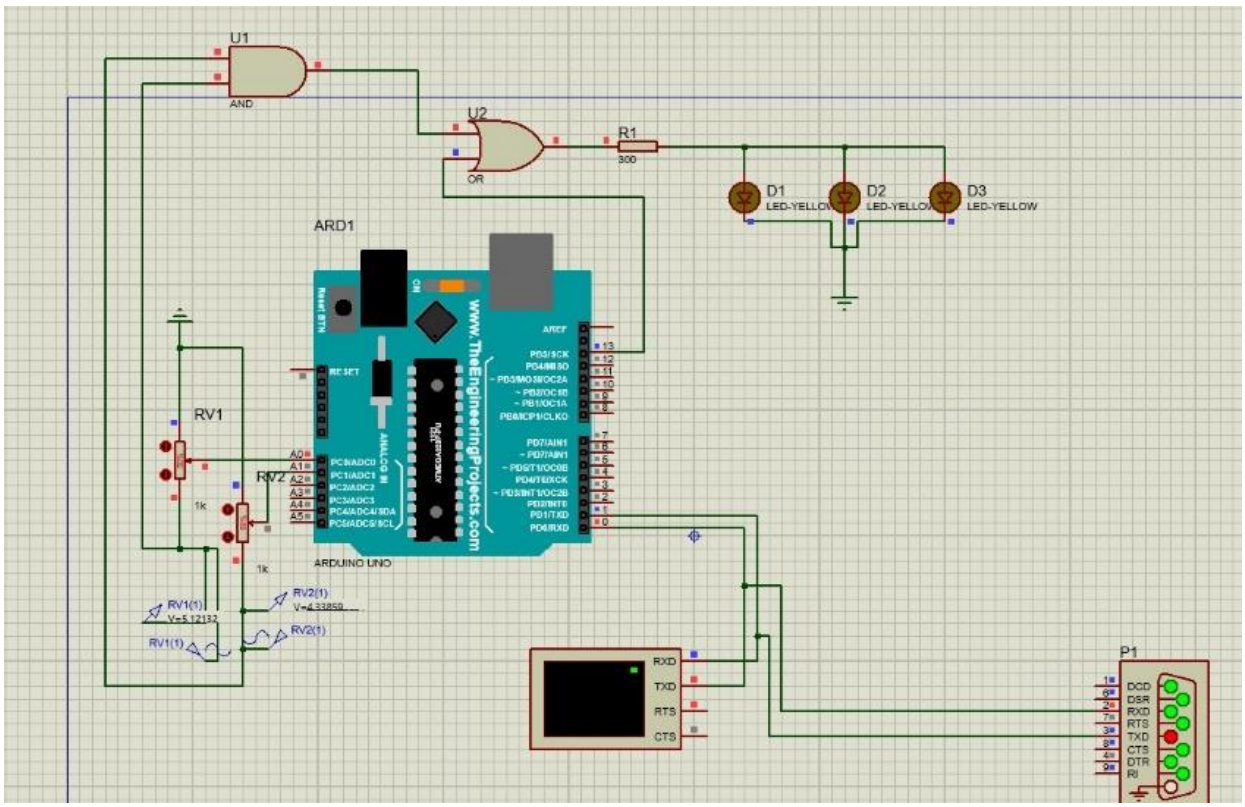
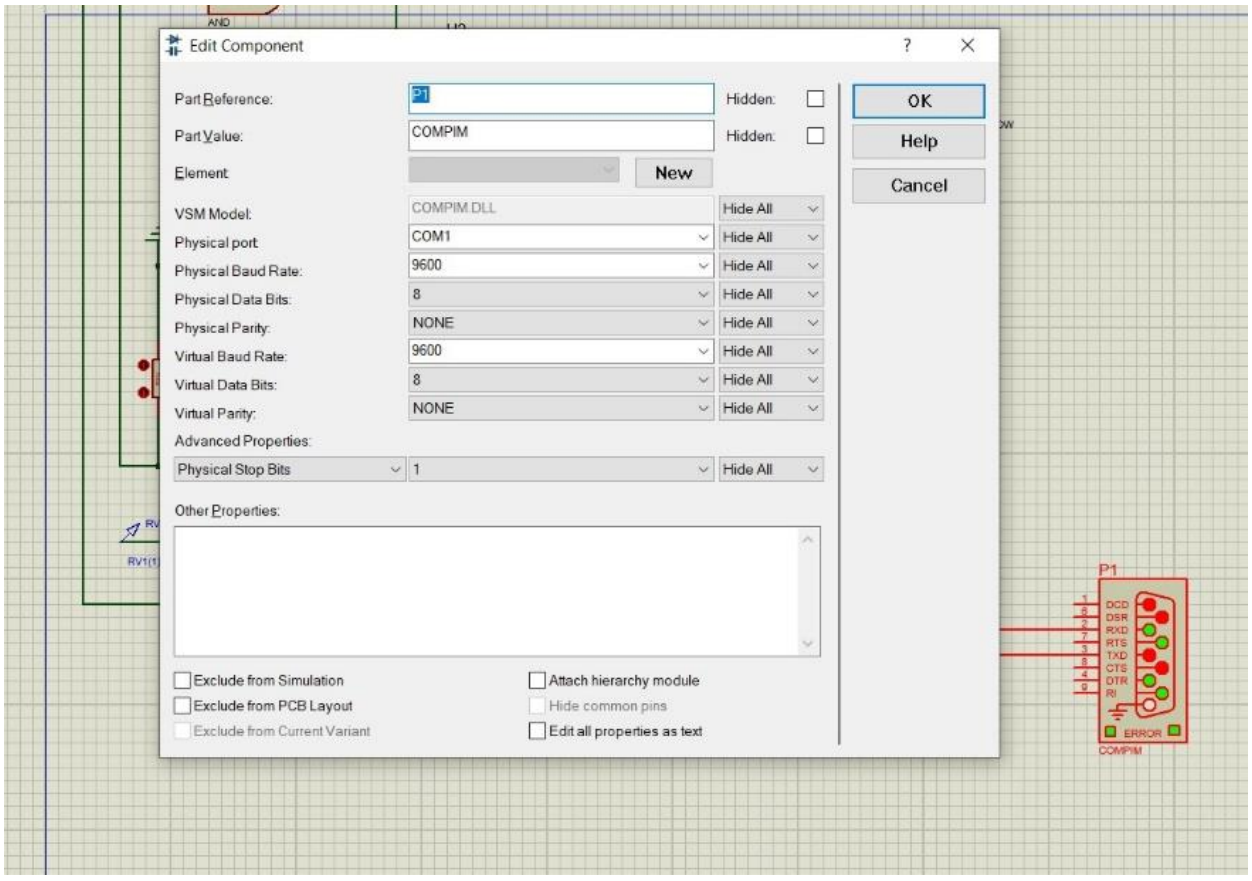
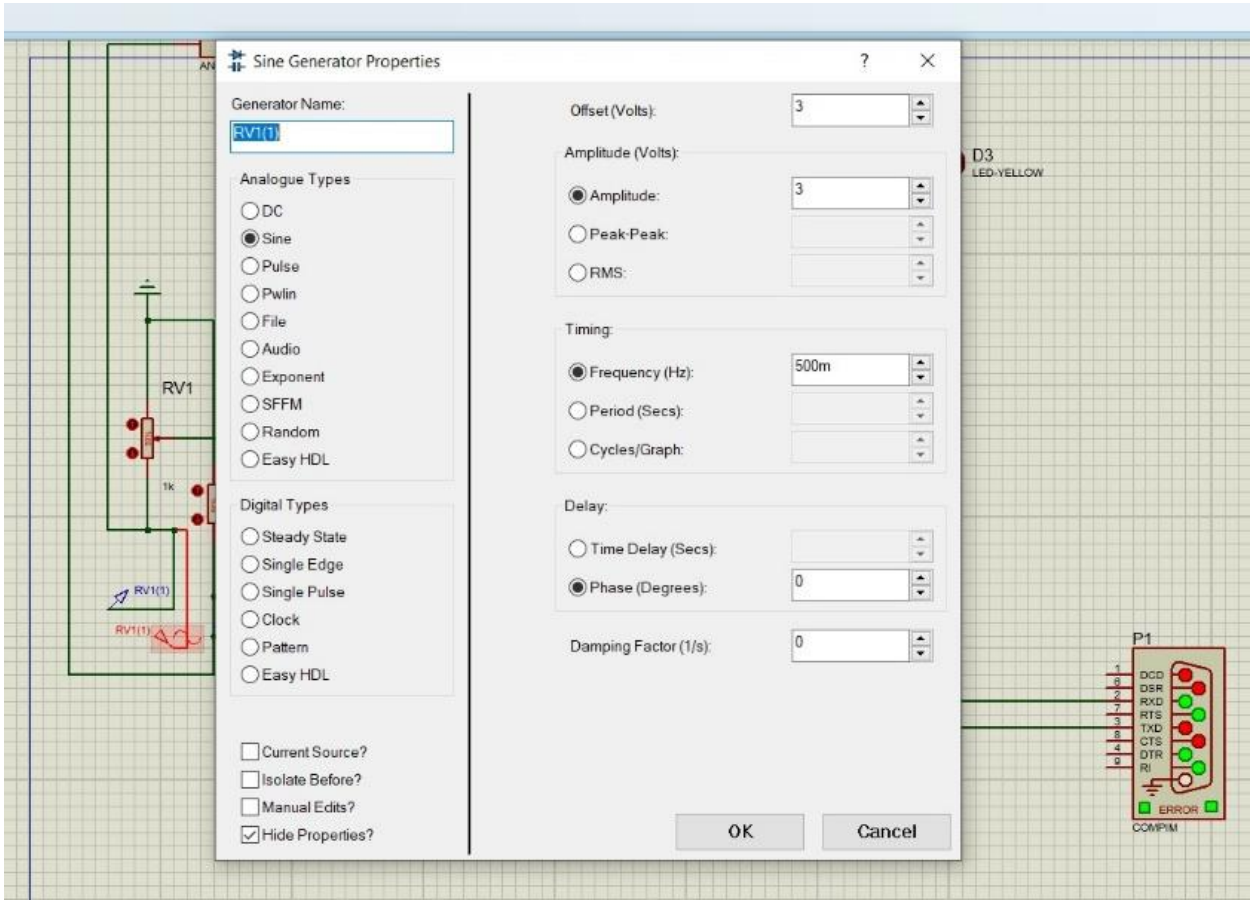
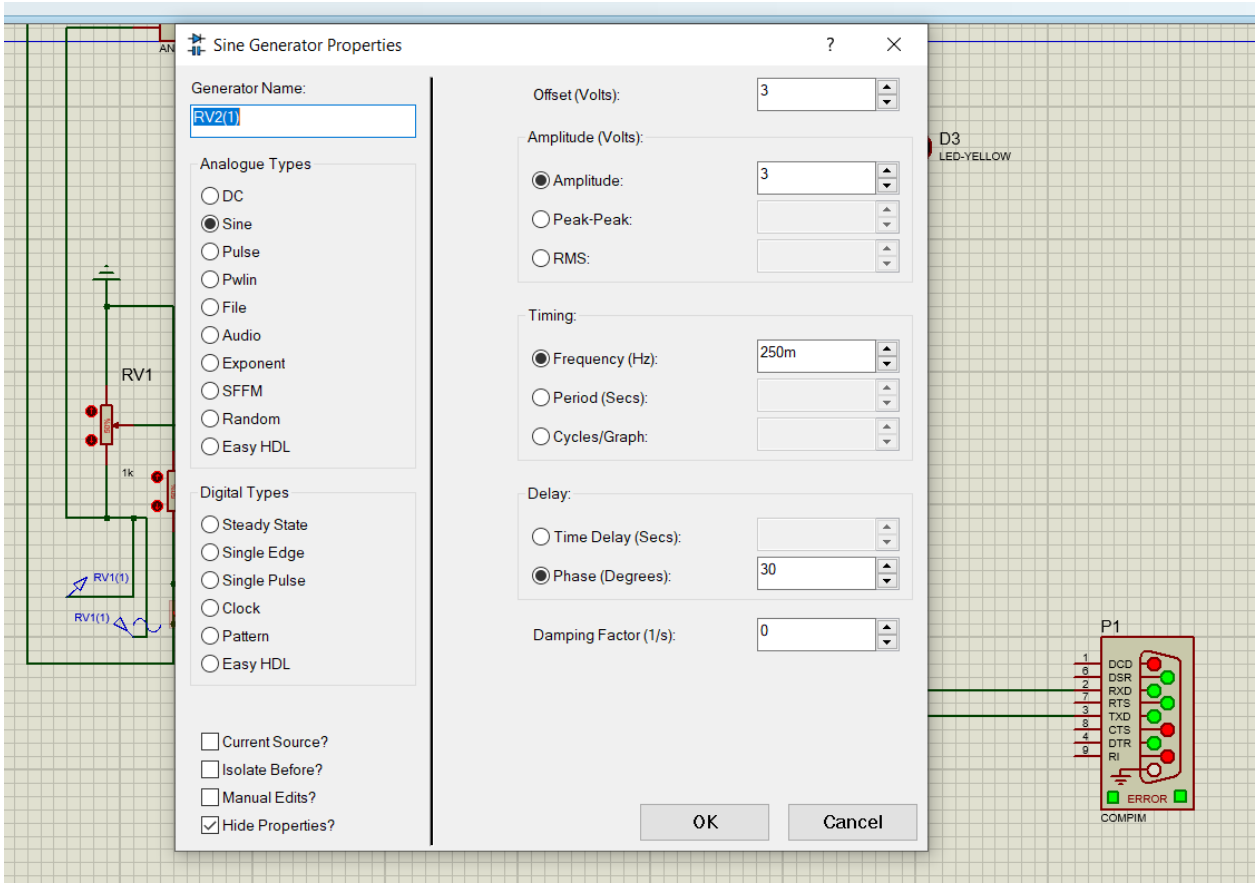


Рисунок 2 Плата не подает питание, когда генераторы работают стабильно







Выше приведены настройки COMPIM, первого и второго генераторов.

Заключение:

На примере работы данной схемы я показал механизм аналогичных систем более крупного масштаба, которые также могут регулировать аварийную подачу питания на важные инфраструктурные объекты. Плата Arduino однако не может принимать на вход и выдавать напряжение выше 5 вольт, однако симуляция работы освещения для неё под силу.

Список источников:

- Официальная документация Arduino https://docs.arduino.cc/hardware/uno-rev3

- Документация Scikit-learn https://scikit-learn.org/stable/

- Документация Virtual Serial Port Driver https://www.virtual-serial-port.org