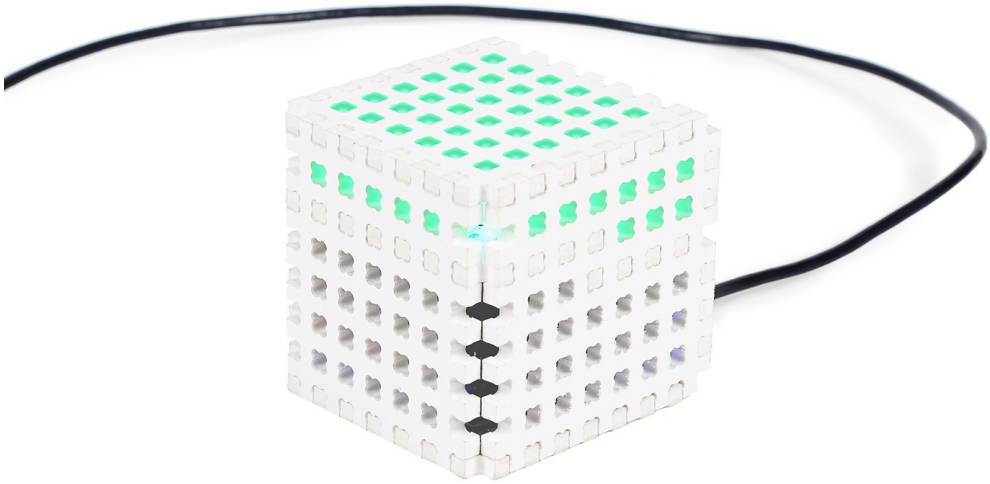
Технокуб

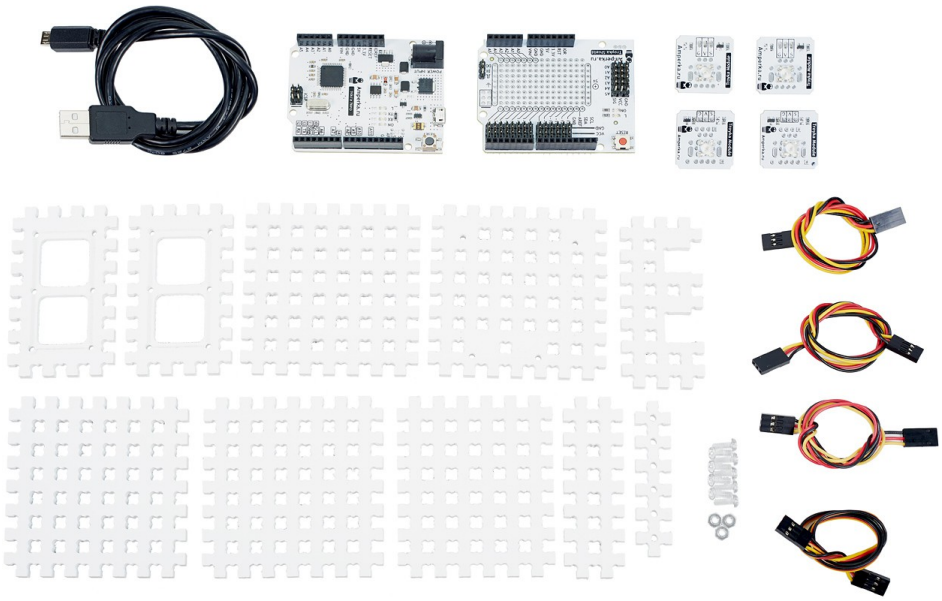


- Платформы: Iskra Neo
- Языки программирования: Arduino (C++), Python
- Тэги: куб, конструктор, мониторинг почты, нотификатор почты, firmata.

Что это?

Ждёте новое сообщение на почту, но не хотите каждый раз отвлекаться, обновлять страницу и смотреть в монитор?! Тогда наше умное и одновременно дизайнерское устройство Технокуб для вас. Суть его довольна проста: пока у вас нет непрочитанных сообщений он горит красным, но как только на почте появляется что-то новенькое — мигает и загорается зелёным. Корпус куба создан из специального, разработанного нами, конструктора на основе ПВХ.

Что нам понадобится?

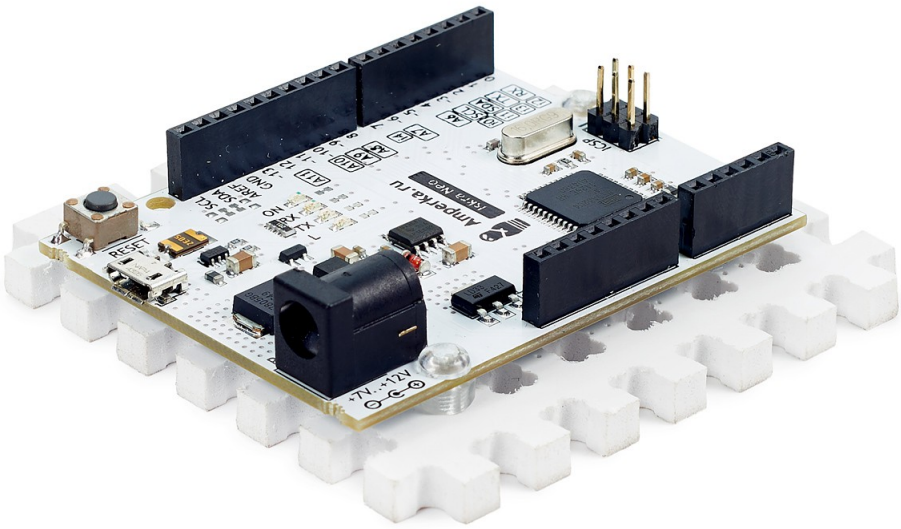


1. Iskra Neo
2. Troyka Shield
3. Светодиод «Пиранья» (Troyka-модуль) красный 2 шт. и зелёный 2 шт.
4. Кабель USB (A — Micro USB)
5. Крепёжные элементы: винты и гайки
6. Конструктор ПВХ

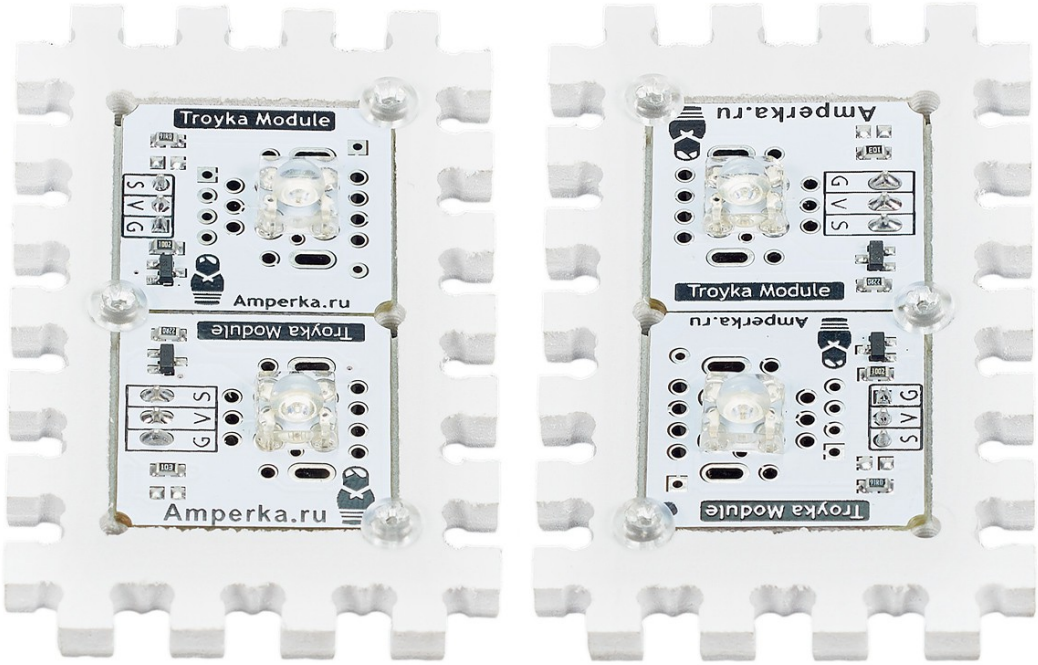
Всё необходимое для сборки, включая детали конструктора, мы собрали в специальный набор

Как собрать?

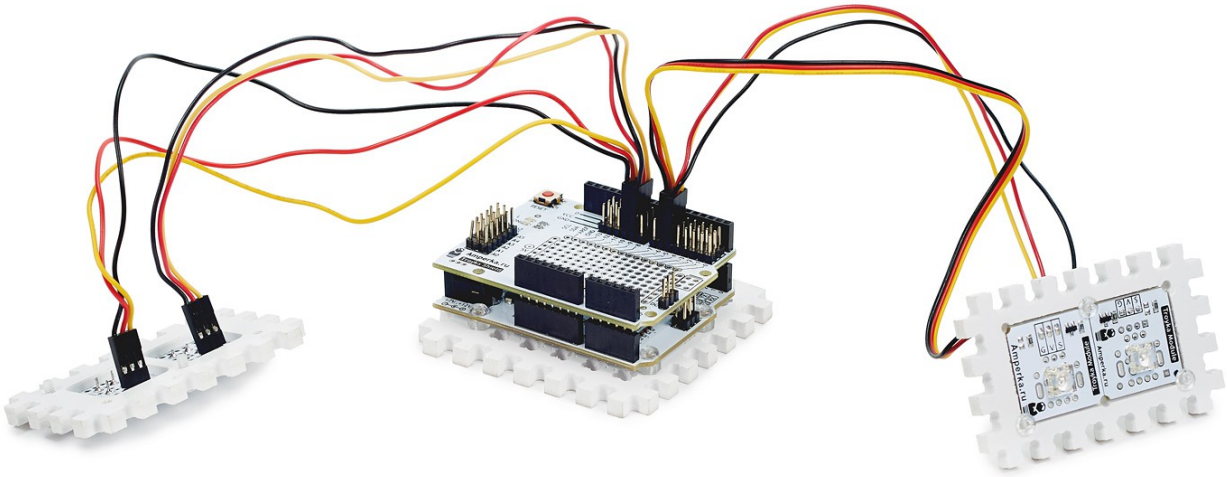
1. Возьмите платформу Iskra Neo и нижнюю панель куба (7x7), соедините её с помощью винтов и гаек, так чтобы гайки располагались между панелью и платформой.



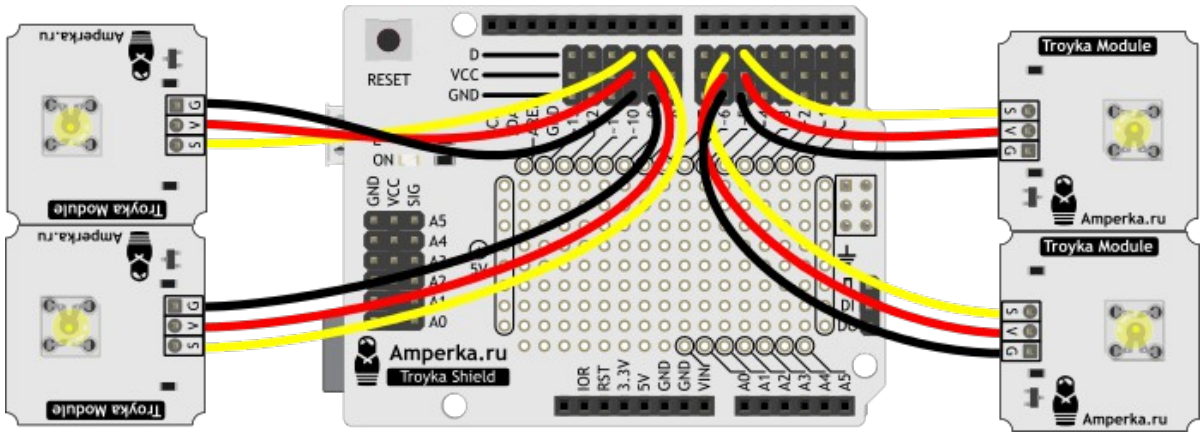
2. Далее возьмите два светодиода «Пиранья» ,красный и зелёный, прикрутите их к панели для крепления двух Troyka-модулей. Повторите процесс с двумя оставшимися светодиодными модулями и второй панелью для крепления двух Troyka-модулей.



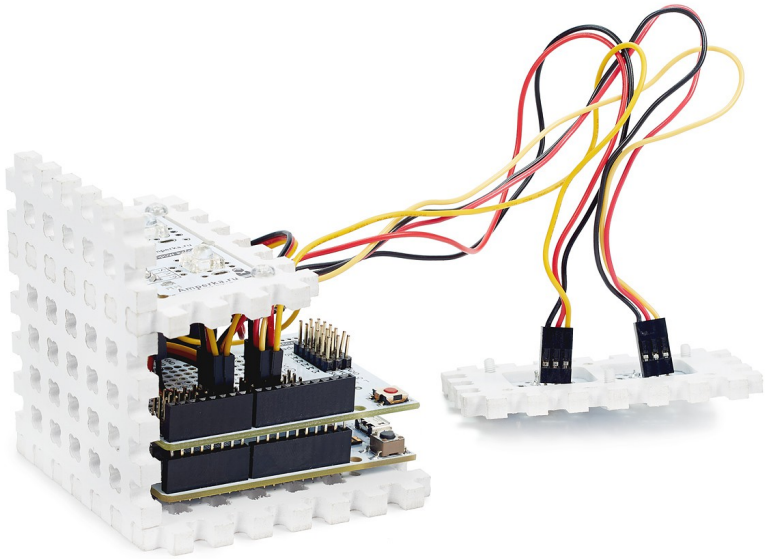
3. Установите Troyka Shield на платформу Iskra Neo. Подключите через стандартные 3-проводные шлейфы к Troyka Shield одну пару светодиодных модулей (дальняя панель светодиодов): красный к 5 пину, зелёный к 6 пину. Вторую пару (ближняя панель светодиодов): красный к 9 пину , зелёный к 10 пину.



В итоге должна получиться схема, как на рисунке ниже.

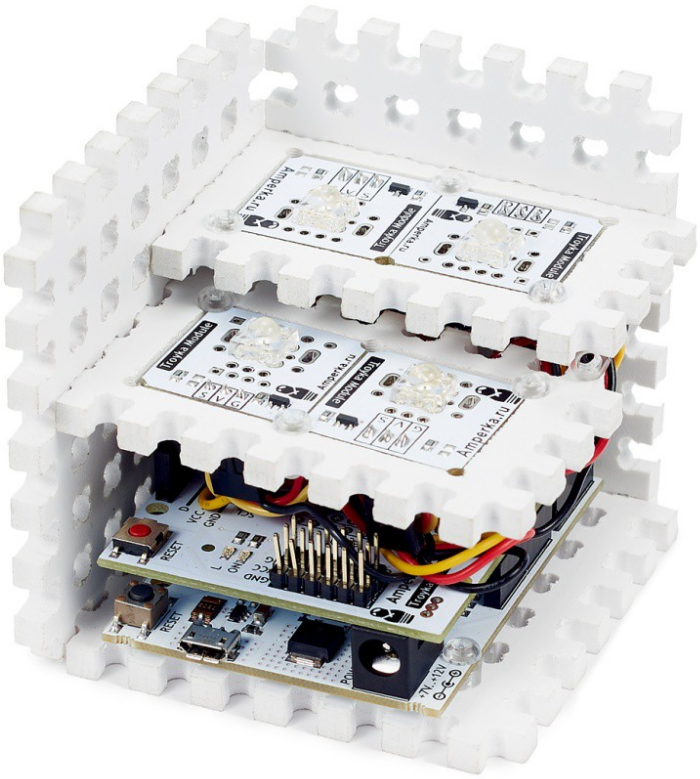
4. 

5. Соедините заднюю панель куба (6x7) с нижней панелью куба (7x7) и дальней панелью со светодиодами, укладывая 3-проводные шлейфы между Troyka Shield и дальней панелью со



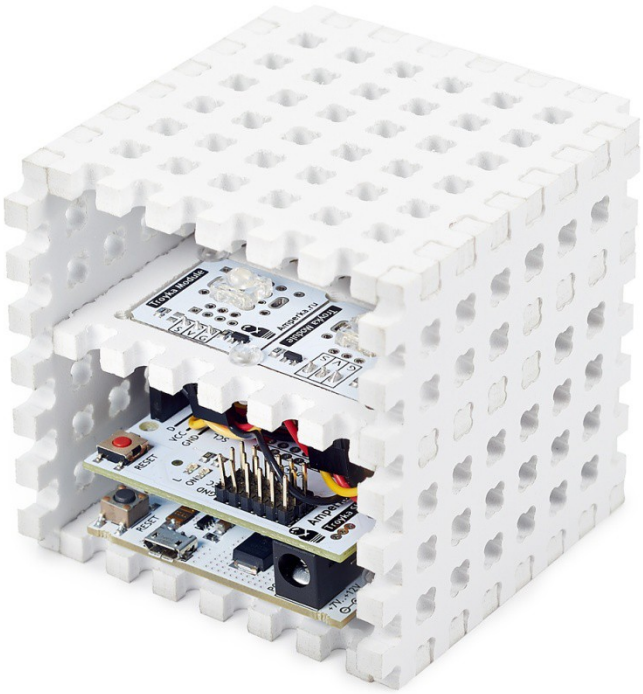
светодиодами
6. Возьмите левую боковую панель куба (7x7), соедините её с нижней панелью куба (7x7) и с дальней панелью со светодиодами. Также возьмите ближнюю панель со светодиодами и соедините её с левой боковой панелью куба (7x7), укладывая 3-проводные шлейфы между Troyka Shield и

ближней панелью со светодиодами

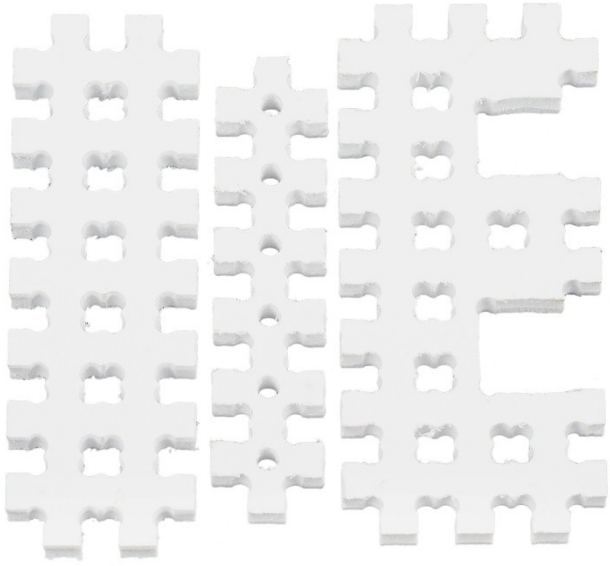


7. Теперь возьмите правую панель куба (7x7) и соедините её с нижней панелью куба (7x7), а также с ближней и дальней панелью со светодиодами. После этого установите верхнюю панель

(7x7) на левую, правую и заднюю панели куба.

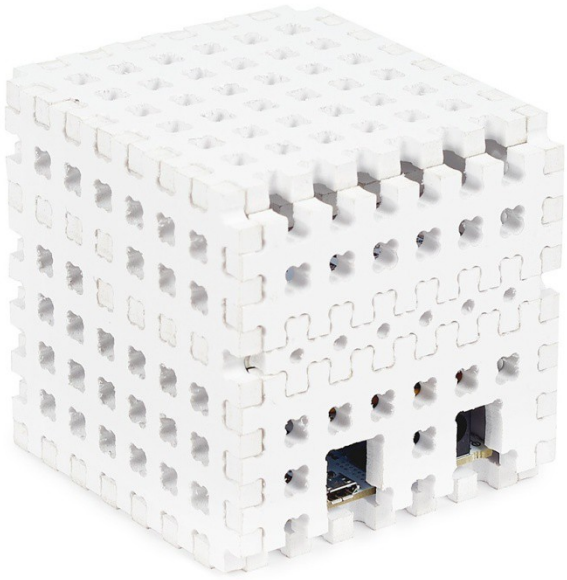


8. Соедините переднюю панель куба (7x2) с передней панелью (7x3) через панель крестиков

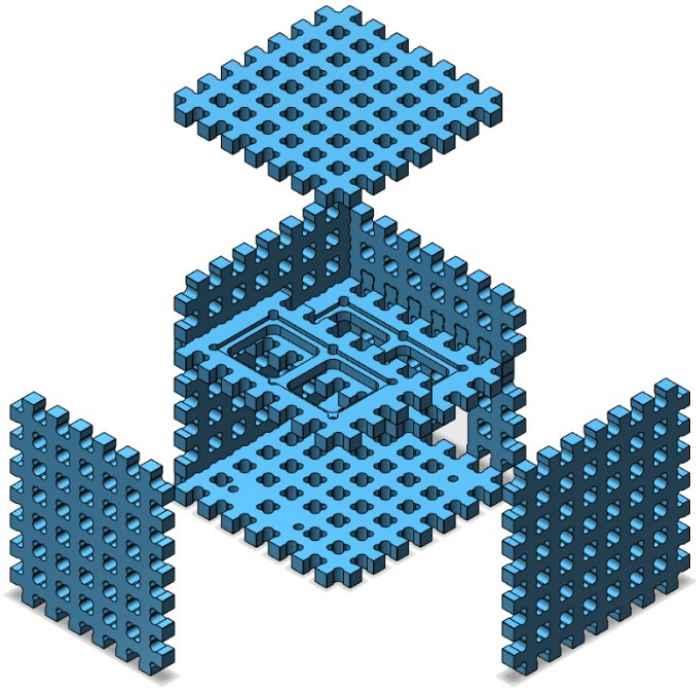


(1x6).

9. Закрепите собранную панель из предыдущего пункта с левой и правой панелью куба.



Для проверки правильности сборки куба рекомендуем воспользоваться сборочным чертежом.



Как запустить?

1. Установите и настройте платформу Iskra в ОС Windows. Помните платформа Iskra Neo является эквивалентом итальянской Arduino Leonardo, поэтому везде при выборе платформы выбирайте Arduino Leonardo.
2. Подготовьте платформу Iskra Neo для работы с приложениями из компьютера. Для этого прошейте её специальной прошивкой Firmata, которую вы можете найти среди стандартных примеров из библиотеки Firmata: Файл Образцы Firmata *StandartFirmata* и загрузить в платформу Iskra Neo, как обычный скетч.
3. Далее скачайте и установите программу для управления кубом на компьютер.

Скачать Cube

Доработка приложения для Windows

Данный раздел о том, как внести изменения в программу и сделать её совместимой с Windows. Если вы не собирается вносить изменения в логику работы программы, можете пропустить этот раздел.

Протокол Firmata

Связь между платформой Iskra Neo и программой на компьютере идёт через стандартный протокол Firmata. Любое программное обеспечение на любом компьютере, которое совместимо с Serial-соединением, может обмениваться данными с микроконтроллером, используя Firmata.

Установка среды разработки

Для создания и программирования приложения воспользуемся языком программирования Python.
Скачать Python 2.7.9

Помимо самой среды, необходимо установить wxPython — библиотеку графического интерфейса пользователя для Python.
Скачать wxPython 3.0 для Python 2.7
для 32-битного Windows
для 64-битного Windows

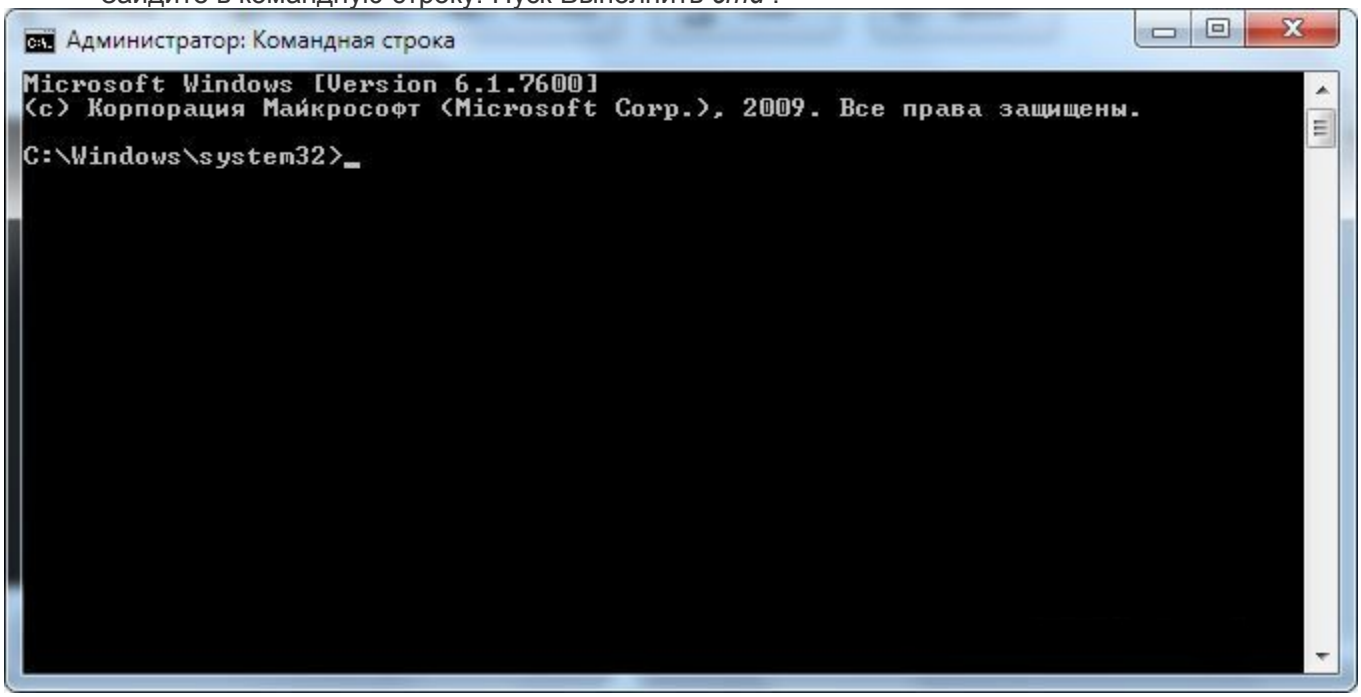
Создание проекта

Скачайте проект и поместите его содержимое в отдельную директорию на жёстком диске, допустим:*C:\Projects\Cube*\. Проект состоит из файлов и директорий:

- main.py — графический пользовательский интерфейс;
- device.py — управление светодиодами на низком уровне;
- modes.py — файл, который отвечает за проверку почты
- setup.py — файл, для конвертации приложения под Windows.
- icons.py — информация об иконках
- директория icons — в этой директории, хранятся иконки программы.
- .gitignore — файл для работы с Git — Запись изменений в репозиторий.

Также в нашем проекте мы воспользуемся готовой библиотекой pyfirmata для Python. Её необходимо установить вручную:

1. Зайдите в командную строку: Пуск Выполнить *cmd* :



2. Используя команды командной строки перейдите в директорию, куда вы поместили исходные файлы проекта *C:\Projects\Cube*
3. Допишите такую строку C:\Python27\Scripts\pip.exe install pyfirmata и дождитесь сообщения об удачной установке.

Пробный запуск

Теперь попробуем запустить проект. Оставаясь в командной строке, в директории с проектом введите строку main.py. Если вы сделали всё правильно, должна запуститься форма с пользовательским интерфейсом.

Теперь вы смело можете изменять и редактировать код программы. Например в файле main.py классManualControlPanel отвечает за графический интерфейс программы, когда выбран ручной режим.

main.py

```
class ManualControlPanel(wx.Panel):
    def __init__(self, parent, device):
        super(ManualControlPanel, self).__init__(parent)
        self.device = device
        self.InitUI()

    def InitUI(self):
        self.red_button = wx.Button(self, label=u'Красный')
        self.green_button = wx.Button(self, label=u'Зелёный')

        self.red_button.Bind(wx.EVT_BUTTON, self.OnRedButton)
        self.green_button.Bind(wx.EVT_BUTTON, self.OnGreenButton)

        box = wx.BoxSizer(wx.HORIZONTAL)
        box.Add(self.red_button, 1, wx.EXPAND | wx.BOTTOM, 10)
        box.Add(self.green_button, 1, wx.EXPAND | wx.BOTTOM, 10)
        self.SetSizer(box)
```

```
    def OnRedButton(self, event):
        self.device.go_red()

    def OnGreenButton(self, event):
        self.device.go_green()

    def ActivateMode(self):
        pass

    def DeactivateMode(self):
        pass
```

А в файле modes.py класс ImapMode, отвечает за логику программы проверки новых писем на сервере IMAP.

modes.py

```
class ImapMode(Mode):
    def __init__(self, device, interval=20):
        super(ImapMode, self).__init__()
        self.device = device
        self.interval = interval
        self.status = u''
        self._prev_count = 0
        self._host = None
        self._port = None
        self._login = None
        self._password = None

    def set_host_port(self, host, port):
        self._host = host
        self._port = port

    def set_credentials(self, login, password):
        self._login = login
        self._password = password

    def loop(self):
        self._stopped = False
        while not self._stopped:
            self.set_status(u"Проверка почты…")
            count = 0

            try:
                count = self._fetch_unread_count()
                message = u"Писем: {}".format(count)
            except imaplib.IMAP4.error as e:
                message = u"Неверные логин/пароль"
            except socket.error:
                message = u"Нет соединения с сервером"

            self.set_status(message)
```

```
            if self._stopped:
                break

            if count > self._prev_count:
                self.device.blink()

            if count:
                self.device.go_green()
            else:
                self.device.go_red()

            self._prev_count = count

            countdown = self.interval
            while countdown > 0 and not self._stopped:
                sleep(0.1)
                countdown -= 0.1
                self.set_status(u"{} ~ {:.0f}".format(message, countdown))

    def stop(self):
        self._stopped = True

    def set_status(self, status):
        self.status = status
        self._post_event(StatusChangedEvent(status=status))

    def _fetch_unread_count(self):
        connection = imaplib.IMAP4_SSL(self._host, self._port)
        connection.login(self._login, self._password)
        connection.select()
        resp = connection.search(None, 'UnSeen')
        return len(resp[1][0].split())
```

Запуск программы в Windows

После того, как вы создали или изменили программу на языке Python, встаёт вопрос, как же её запустить на Windows, так как не у всех пользователей Microsoft установлен интерпретатор python с нужными библиотеками. Для этого можно использовать приложение py2exe, которое позволяет упаковать программу на python в .exe файл и кучу полезного хлама, после чего она будет запускаться на любой windows-машине. Для этого необходимо проделать ряд манипуляций:

1. Скачайте и установите приложение py2exe

Скачать py2exe
для 32-битного Windows
для 64-битного Windows

2. Найдите у себя на компьютере в системной директории *c:\windows\system32* библиотекуMSVCP90.dll и скопируйте её в директорию DLLs, которая находиться в директории с установленным Python. По умолчанию: *c:\Python27\DLLs*.
3. Зайдите через командную строку в директорию с проектом, введите строку:setup.py py2exe и дождитесь сообщение об удачной операции.

Если вы всё сделали правильно, в директории с проектом должны были появиться две новых директории:

- build — служебная, её можно сразу удалить.
- dist — собственно, в ней лежит наша программа с файлом .exe для запуска.

Алгоритм

- Сразу после запуска приложения, программа проверяет количество подключенных COM-портов.
  - Если больше одного, то программа предложит выбрать один из них.
  - Если один — сразу подключится к нему автоматически.
  - Если ни одного — сообщит об ошибке и предложит повторить попытку.
- Выбор режима работы приложения (бесконечный цикл):
  - Ручное управление
    - Кликнув на первую кнопку загораются красные светодиоды.
    - Кликнув на вторую кнопку загораются зеленные светодиоды.
  - Проверка почты на GMail / Mail
    - Вводим данные: Логин/пароль и подтверждаем.
    - Если писем нет, горит красный светодиод.
    - Если есть новое письмо, мигаем попеременно светодиодами в течении 3 секунд и оставляем гореть зеленый.
  - Проверка почты через сервер IMAP
    - Вводим данные: Логин, пароль, сервер IMAP и подтверждаем.
    - Если писем нет, горит красный светодиод.
    - Если есть новое письмо, мигаем попеременно светодиодами в течении 3 секунд и оставляем гореть зеленый.