Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный
университет»

УТВЕРЖДАЮ	СОГЛАСОВАНО
Декан ФКТ	Заведующий кафедрой ПМИ
_____Я.Ю. Григорьев	_____ С.А. Гордин
«____» _____ 2019 г.	«____» _____ 2019 г.

Программное обеспечение для управления освещением

Руководитель СКБ Е.П. Жарикова

Подпись/дата

Ответственный исполнитель О.В. Попова

Подпись/дата

Комсомольск-на-Амуре 2019

Карточка проекта

Название	Программное обеспечение для управления освещением
Тип проекта	Инициативный (инициативный, по заказу, в рамках конкурса, учебная работа, другое)
Исполнители	Попова О.В. – 6ВСб-1 ответственный исполнитель
Срок реализации	10.2019-12.2019 Месяц, год

Использованные программные средства

Наименование	**Версия**
ubuntu	**16.04 LTS**
Arduino IDE	**2:1.0.5**
Python	**3.8**
ROS	**kinetic**

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ЗАДАНИЕ

на разработку

Выдано студентам:
Поповой О. В. – 6ВСб-1

Название проекта:

Программное обеспечение для управления освещением

Назначение: Программное обеспечение предназначенное для управлением светом пользователем посредством жестов и речи.

Применение: Программное обеспечение может применяться для управления светом.

Функциональное описание Программного обеспечения:

Пользователь произносит кодовое слово «подручный», что активирует прослушивание фразы, далее пользователь говорит, что он хочет сделать со светом. Также пользователь может показать необходимые жесты для включения или выключения света.

Требования:

Данное ПО должно точно распознавать голосовые команды и жесты

План работ:

Этап	Дата начала	Дата окончания
Формирование требований к программному обеспечению	01.10.19	03.10.19
Разработка структуры программного обеспечения	04.10.19	10.10.19
Разработка и утверждение технического задания	11.11.19	16.11.19
Программная реализация	17.11.19	01.12.19
Тестирование и отладка системы	02.12.19	10.12.19
Подготовка документации	11.12.19	13.12.19
Опытная эксплуатация	14.12.19	20.12.19

Руководитель СКБ Е.П. Жарикова

Подпись/дата

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный
университет»

Руководство администратора

Программное обеспечение для управления освещением

Руководитель СКБ Е.П. Жарикова

Подпись/дата

Ответственный исполнитель О.В. Попова

Подпись/дата

Комсомольск-на-Амуре 2019

Содержание

1 Общие положения

Настоящий документ представляет собой руководство администратора программного обеспечения «Смарт лайт» далее ПО.

Руководство определяет порядок установки, настройки и администрирования ПО.

Перед установкой и эксплуатацией системы рекомендуется внимательно ознакомиться с настоящим руководством.

Документ подготовлен в соответствии с РД 50-34.698-90 - в части структуры и содержания документов, и в соответствии с ГОСТ 34.201-89 - в части наименования и обозначения документов.

1.1 Наименование программы

Наименование программного продукта: программное обеспечение для управления освещением «Смарт лайт».

1.2 Наименования документов, на основании которых ведется проектирование системы

Создание ПО осуществляется на основании требований и положений следующих документов:

- задание.

1.3 Перечень организаций, участвующих в разработке системы

Разработчики: студент группы 6ВСб-1 Комсомольского-на-Амуре государственного технического университета О. В. Попова.

Заказчик: Комсомольский-на-Амуре государственный университет, студенческое конструкторское бюро «Интеллектуальные технологии».

1.4 Сведения об использованных при проектировании нормативно-технических документах

При проектировании использованы следующие нормативно-технические документы:

- ГОСТ 19.101-77 – виды программ и программных документов;

- ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом

- ГОСТ 19.201-78 Техническое задание, требования к содержанию и оформлению;

- ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению

- ГОСТ 2.004-88. Единая система конструкторской документации. Общие требования к выполнению конструкторских технологических документов на печатающих и графических устройствах вывода ЭВМ.

2 Назначение и принцип действия

2.1 Назначение изделия

Программное обеспечение для управления светом – предназначено для регулировки уровня освещения в помещении.

Для полноценного использования программного обеспечения, из дополнительного оборудования, необходимы: микрофон, камера, светодиод.

2.2 Области использования изделия

Программное обеспечение может применяться для управления светом.

2.3 Принцип действия

Пользователь произносит кодовое слово «подручный», что активирует прослушивание фразы, далее пользователь говорит, что он хочет сделать со светом. Также пользователь может показать жесты: (один палец) для включения и (два пальца) выключения света.

3 Описание программного обеспечения

3.1 Описание работы программного обеспечения

Если запущен файл для распознавания жестов, то откроется окно с выделенной областью куда нужно что бы попал жест(Рисунок 3.1), и после того как будет распознан жест, будет выполнено действие включения или выключения светодиода.

Если запущен файл для распознавания голосовых команд, то следует проговорить в микрофон ключевое слово «подручный» и то действие которое включит или выключит свет. После этого будет выполнено действие включения или выключения светодиода. Так же программа «ответит» о выполненном действии «включил» или «выключил».

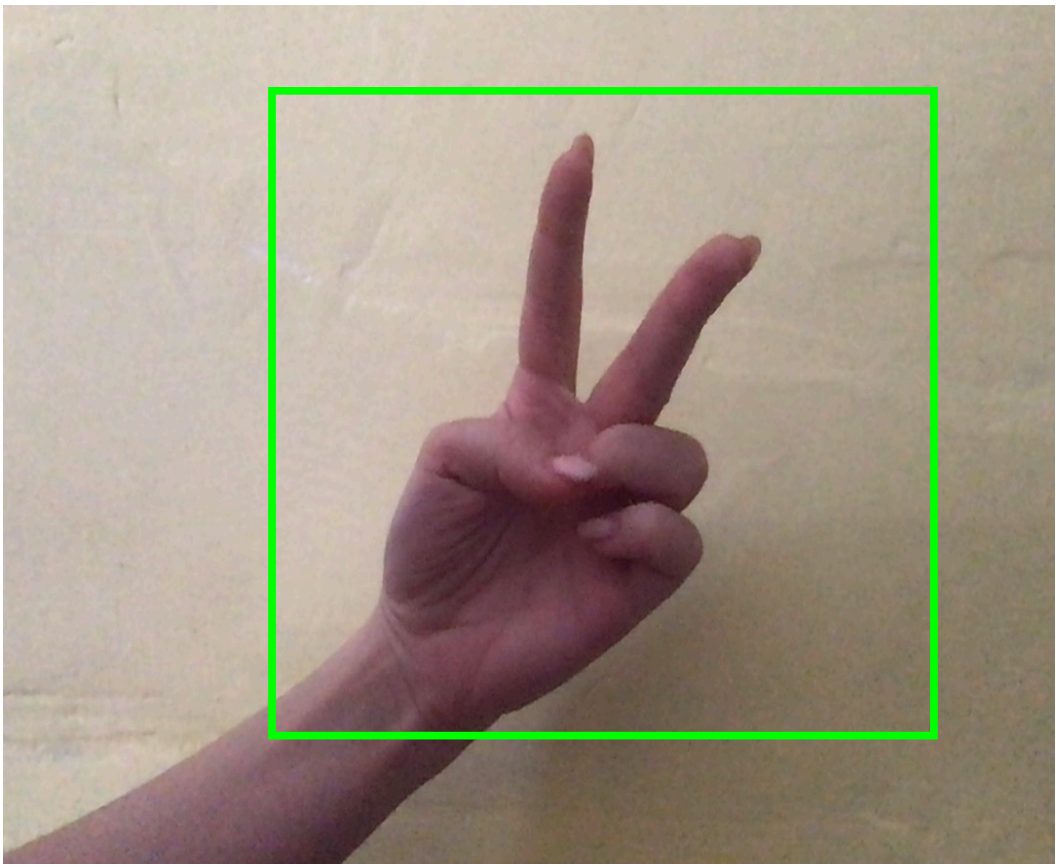

Рисунок 3.1 – область распознавания жестов

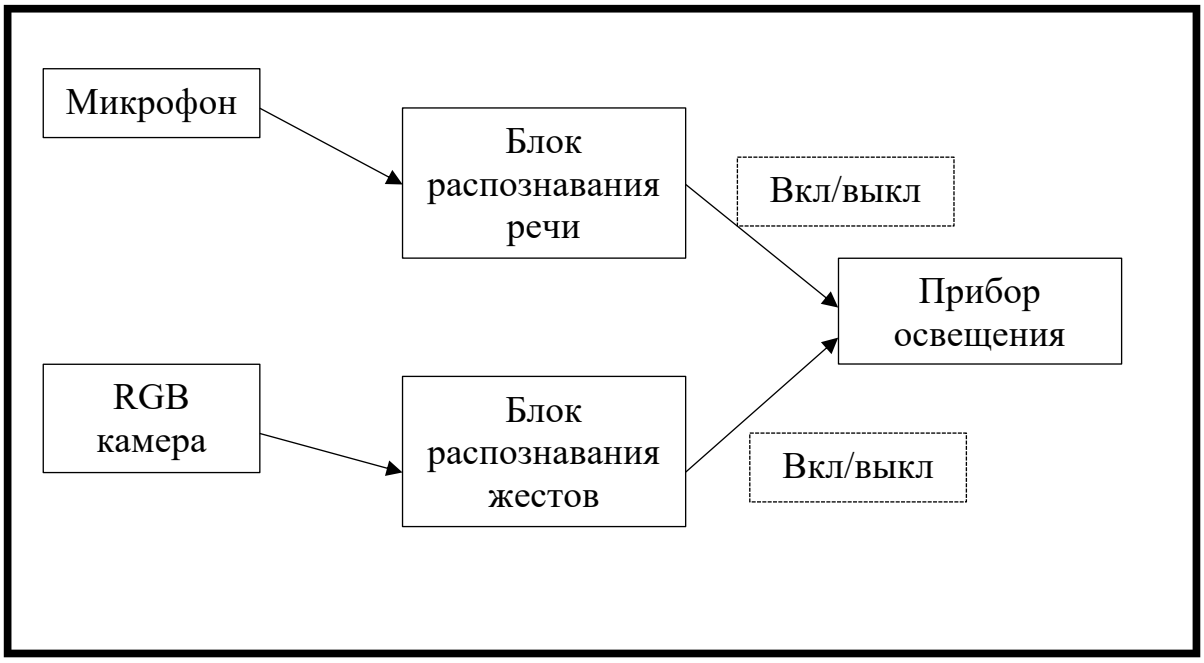

Рисунок 3.2 – схема работы ПО

3.3 Используемые технические средства

Техническое обеспечение необходимое для работы программы:

- оперативная память 1 Гб и выше;
- двуядерный процессор Intel Core i3 2 ГГц;
- существование логических и/или физических дисков со свободным дисковым пространством 100 Мб и выше;
- камера
- микрофон

3.4 Вызов и загрузка

Для функционирования программного обеспечения на персональном компьютере должна быть установлена операционная система Ubuntu с 16.04 по 19.04. Перед началом использования изделия необходимо установить камеру на ровную поверхность, фон(стена) должна быть однородного цвета. Подключить

микрофон и камеру через USB. Отойти от камеры на расстояние не меньше 0,5 м.

Включить проект recognize.py или speech_to_cmd.py командами в терминалах:

1. roscore

2. rosrun (speech_to_cmd.py или recognize.py)

3. rosrun rosserial_python serial_node.py/dev/ttyACM0 (порт может блыть другим. Название порта определяется командой в терменале ls /dev/tty*)

3.5 Входные данные

Пользователь показывает жесты или произносит фразы для управления светом.

3. 6 Выходные данные

Выходными данными программного обеспечения для управления светом является создание управляющего сигнала (вкл/ выкл).

4. ТЕКСТ ПРОГРАММЫ

Программа для распознавания жестов

```
  #!/usr/bin/env python
# *-* coding: utf-8 *-*
#--------------------------------------------------------------
# SEGMENT, RECOGNIZE and COUNT fingers from a video sequence
#--------------------------------------------------------------

# organize imports
import cv2
import os
import rospy
from std_msgs.msg import Bool
import imutils
import numpy as np
from sklearn.metrics import pairwise

# global variables
bg = None

#-------------------------------------------------
# To find the running average over the background
#-------------------------------------------------
def run_avg(image, accumWeight):
    global bg
    # initialize the background
    if bg is None:
        bg = image.copy().astype("float")
        return

    # compute weighted average, accumulate it and update the background
    cv2.accumulateWeighted(image, bg, accumWeight)

#-----------------------------------------------
# To segment the region of hand in the image
#-----------------------------------------------
def segment(image, threshold=25):
    global bg
    # find the absolute difference between background and current frame
    diff = cv2.absdiff(bg.astype("uint8"), image)

    # threshold the diff image so that we get the foreground
    thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)[1]

    # get the contours in the thresholded image
    (_, cnts, _) = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # return None, if no contours detected
    if len(cnts) == 0:
        return
```

```
    else:
        # based on contour area, get the maximum contour which is the hand
        segmented = max(cnts, key=cv2.contourArea)
        return (thresholded, segmented)

#--------------------------------------------------------------
# To count the number of fingers in the segmented hand region
#--------------------------------------------------------------
def count_fingers(thresholded, segmented):
    # find the convex hull of the segmented hand region
    chull = cv2.convexHull(segmented)

    # find the most extreme points in the convex hull
    extreme_top    = tuple(chull[chull[:, :, 1].argmin()][0])
    extreme_bottom = tuple(chull[chull[:, :, 1].argmax()][0])
    extreme_left   = tuple(chull[chull[:, :, 0].argmin()][0])
    extreme_right  = tuple(chull[chull[:, :, 0].argmax()][0])

    # find the center of the palm
    cX = int((extreme_left[0] + extreme_right[0]) / 2)
    cY = int((extreme_top[1] + extreme_bottom[1]) / 2)

    # find the maximum euclidean distance between the center of the palm
    # and the most extreme points of the convex hull
    distance = pairwise.euclidean_distances([(cX, cY)], Y=[extreme_left, extreme_right,
extreme_top, extreme_bottom])[0]
    maximum_distance = distance[distance.argmax()]

    # calculate the radius of the circle with 80% of the max euclidean distance obtained
    radius = int(0.8 * maximum_distance)

    # find the circumference of the circle
    circumference = (2 * np.pi * radius)

    # take out the circular region of interest which has
    # the palm and the fingers
    circular_roi = np.zeros(thresholded.shape[:2], dtype="uint8")

    # draw the circular ROI
    cv2.circle(circular_roi, (cX, cY), radius, 255, 1)

    # take bit-wise AND between thresholded hand using the circular ROI as the mask
    # which gives the cuts obtained using mask on the thresholded hand image
    circular_roi = cv2.bitwise_and(thresholded, thresholded, mask=circular_roi)

    # compute the contours in the circular ROI
    (_, cnts, _) = cv2.findContours(circular_roi.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)

    # initalize the finger count
    count = 0

    # loop through the contours found
    for c in cnts:
```

```
            # compute the bounding box of the contour
            (x, y, w, h) = cv2.boundingRect(c)

            # increment the count of fingers only if -
            # 1. The contour region is not the wrist (bottom area)
            # 2. The number of points along the contour does not exceed
            #     25% of the circumference of the circular ROI
            if ((cY + (cY * 0.25)) > (y + h)) and ((circumference * 0.25) > c.shape[0]):
                count += 1

        return count

    #-----------------
    # MAIN FUNCTION
    #-----------------
    if __name__ == "__main__":
        # initialize accumulated weight
        accumWeight = 0.5

        # get the reference to the webcam
        camera = cv2.VideoCapture(0)

        # region of interest (ROI) coordinates
        top, right, bottom, left = 10, 350, 225, 590

        # initialize num of frames
        num_frames = 0

        # calibration indicator
        calibrated = False


        rospy.init_node('geasture', anonymous=True)

        light_status = rospy.Publisher("/light_status", Bool, queue_size=10)
        rate = rospy.Rate(10) # 10hz

        # keep looping, until interrupted
        count = 0;
        prev_status = 2;
        sent = False
        while not rospy.is_shutdown():
            # get the current frame
            (grabbed, frame) = camera.read()

            # resize the frame
            frame = imutils.resize(frame, width=700)

            # flip the frame so that it is not the mirror view
            frame = cv2.flip(frame, 1)

            # clone the frame
            clone = frame.copy()
```

```
        # get the height and width of the frame
        (height, width) = frame.shape[:2]

        # get the ROI
        roi = frame[top:bottom, right:left]

        # convert the roi to grayscale and blur it
        gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (7, 7), 0)

        # to get the background, keep looking till a threshold is reached
        # so that our weighted average model gets calibrated
        if num_frames < 30:
            run_avg(gray, accumWeight)
            if num_frames == 1:
                print("[STATUS] please wait! calibrating...")
            elif num_frames == 29:
                print("[STATUS] calibration successfull...")
        else:
            # segment the hand region
            hand = segment(gray)

            # check whether hand region is segmented
            if hand is not None:
                # if yes, unpack the thresholded image and
                # segmented region
                (thresholded, segmented) = hand

                # draw the segmented region and display the frame
                cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))

                # count the number of fingers
                fingers = count_fingers(thresholded, segmented)
                if fingers == 1:
                    if count > 10 and not sent:
                        light_status.publish(True)
                        cv2.putText(clone, str('turn on'), (70, 45), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,0,255), )
                        os.system("echo 'Включил' | RHVoice-client -s Aleksandr+CLB | aplay")
                        sent = True
                    elif prev_status == fingers:
                        count += 1
                        cv2.putText(clone, str('turn on'), (70, 45), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,0,255), 2)

                    else:
                        count = 0
                        sent = False
                        cv2.putText(clone, str('turn on'), (70, 45), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,0,255), 2)


                elif fingers == 2:
                    if count > 10 and not sent:
```

```
                    light_status.publish(False)
                    cv2.putText(clone, str('turn off'), (70, 45), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,0,255), 2)
                    os.system("echo 'Выключил' | RHVoice-client -s Aleksandr+CLB | aplay")
                    sent = True
                elif prev_status == fingers:
                    count += 1
                    cv2.putText(clone, str('turn off'), (70, 45), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,0,255), 2)

                else:
                    count = 0
                    sent = False
                    cv2.putText(clone, str('turn off'), (70, 45), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,0,255), 2)


            # show the thresholded image
            cv2.imshow("Thesholded", thresholded)

        # draw the segmented hand
        cv2.rectangle(clone, (left, top), (right, bottom), (0,255,0), 2)

        # increment the number of frames
        num_frames += 1

        # display the frame with segmented hand
        cv2.imshow("Video Feed", clone)

        # observe the keypress by the user
        keypress = cv2.waitKey(1) & 0xFF

        # if the user pressed "q", then stop looping
        if keypress == ord("q"):
            break

# free up memory
camera.release()
cv2.destroyAllWindows()
```

Программа для расчпознавания голоса

```
#!/usr/bin/env python
# *-* coding: utf-8 *-*

import os,sys,time,random
# import rospkg
import pyaudio
import rospy
import time
import os
```

```
from std_msgs.msg import Bool


try:
    # Python 2.x
    from sphinxbase import Config
    from pocketsphinx import Decoder
except ImportError:
    # Python 3.x
    from pocketsphinx.pocketsphinx import *
    from sphinxbase.sphinxbase import *



if __name__ == '__main__':

    rospy.init_node('speech', anonymous=True)

    light_status = rospy.Publisher("/light_status", Bool, queue_size=10)
    rate = rospy.Rate(10) # 10hz

    answer_text = {
        'подручный активировать лампочка': True,
        'подручный активировать свет': True,
        'подручный выключить лампочка': False,
        'подручный выключить свет': False,

    }

    config = Decoder.default_config()
    config.set_string('-hmm',
'/home/ardrone01/catkin_ws/src/smart_light/scripts/psx/zero_ru.cd_cont_4000')
    config.set_string('-jsgf', '/home/ardrone01/catkin_ws/src/smart_light/scripts/psx/bb.jsgf')
    config.set_string('-dict', '/home/ardrone01/catkin_ws/src/smart_light/scripts/psx/ru.dic')
    # config.set_string('-hmm',  'psx/zero_ru.cd_cont_4000')
    # config.set_string('-jsgf', 'psx/bb.jsgf')
    # config.set_string('-dict', 'psx/ru.dic')
    # config.set_string('-logfn', '/dev/null')

    phrase = ''

    decoder = Decoder(config)

    p = pyaudio.PyAudio()

    stream = p.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True,
frames_per_buffer=1024)
    stream.start_stream()
    in_speech_bf = True
    decoder.start_utt()
    target_check = False
    goon = True
    while not rospy.is_shutdown():
        buf = stream.read(1024)
        if buf:
            decoder.process_raw(buf, False, False)
```

```
        if decoder.get_in_speech():
            sys.stdout.write('.')
            sys.stdout.flush()

        if decoder.get_in_speech() != in_speech_bf:
            in_speech_bf = decoder.get_in_speech()

            if not in_speech_bf:
                decoder.end_utt()


                try:
                    if  decoder.hyp().hypstr in answer_text.keys():
                        phrase = decoder.hyp().hypstr
                        light_status.publish(answer_text[phrase])
                        if answer_text[phrase]:
                            print("Включаю")
                            os.system("echo 'Включил' | RHVoice-client -s Aleksandr+CLB | aplay")
                        else:
                            print("Выключаю")
                            os.system("echo 'Выключил' | RHVoice-client -s Aleksandr+CLB | aplay")
                except AttributeError:
                    pass

                decoder.start_utt()
    else:
        break
decoder.end_utt()
```

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный
университет»

УТВЕРЖДАЮ	СОГЛАСОВАНО
Декан ФКТ	Заведующий кафедрой ПМИ
_____Я.Ю. Григорьев	_____ С.А. Гордин
«____» _____ 2019 г.	«____» _____ 2019 г.

АКТ

о приемке в эксплуатацию программного обеспечения

для управления светом

г. Комсомольск-на-Амуре « » _____ 2019 г.

Комиссия в составе представителей:
заказчика Е.П. Жариковой – руководителя СКБ ФКТ, С.А. Гордина – Заведующего кафедрой ПМИ, исполнителя О.В. Поповой – 6ВТб-1
составила акт о нижеследующем: «Исполнитель» передает программное обеспечение для управления светом.
Программное обеспечение для управления прошло опытную эксплуатацию с « » _____ по « » _____ 2019г. и признано годным к эксплуатации. Были протестированы все режимы функционирования, отказы системы, а также аварийные отключения по вине системы не наблюдались.

Руководитель СКБ	Ответственный исполнитель
_____/Е.П. Жарикова /	_____ / О.В. Попова/

Таблица учета работ

Дисциплина	Форма учтенной работы (номер ЛР, КП, КР, РГЗ, зачет, итоговая оценка , экзамен)	Преподаватель (дата, ФИО, подпись)	Примечание (ЗУН полученные при выполнении проекта)