ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ

Государственное бюджетное общеобразовательное учреждение города Москвы «Школа № 1103 имени Героя Российской Федерации А.В. Соломатина»

УДК 636.084.7, 681.58, 681.587

КОРМУШКА С СИСТЕМОЙ РАСПОЗНАВАНИЯ ПТИЦ И СОРТИРОВКОЙ КОРМА

Автор:
ученик 10Б ГБОУ Школа №1103
Донцов Илья Игоревич

Руководитель:
Сокур Мария Евгеньевна,
учитель информатики

Москва, 2023

Цель и задачи работы:

Целью работы является создание прототипа «умной» кормушки с системой распознавания птиц и сортировкой корма с целью правильного кормления их в зимний период, а также для привлечения внимание людей к заботе о птицах.

Задачи проекта:

1. Изучить список зимующих птиц Битцевского лесопарка, распределить их на группы и определить примерный рацион для каждой группы.
2. Разработать систему автоматической подачи корма и очистки.
3. Создать систему распознавания птиц с возможностью удаленного просмотра.
4. Разработать 3D-модели кормушки и изготовить прототип изделия.
5. Осуществить тестирование прибора и подобрать оптимальное питание для птиц.

Объект исследования – зимующие птицы.

Предмет исследования – кормление зимующих птиц Битцевского лесопарка.

Методы работы: анализ литературных источников, классификация, моделирование и прототипирование, эксперимент, статистический анализ.

Результаты работы:

Создана умная кормушка, которая отличает и кормит птиц. Фотографии опознанных птиц вместе с их названиями отправляются в телеграмм-канал.

Значимость работы:

При использовании данного изделия прививается любовь к природе, животным, птицам и бережному отношению к природе.

Общий объем пояснительной записки:

Пояснительная записка содержит 15 страниц основного текста, 10 рисунков, 2 таблицы, 3 приложения.

Оглавление

Введение

Как известно, в зимнее время птицы, особенно небольшого размера (воробьи, синицы и т.д.), вынуждены тратить большое количество сил на добычу еды и согревание. Нередко происходит так, что более крупные птицы отбирают еду у уступающих по силе представителей пернатых, а иногда не просто лишают пропитания, но и нападают на них. По этой причине пришла идея создать кормушку, которая будет выдавать корм небольшими порциями и только для определённых видов птиц. Это обеспечит наилучшее пропитание для таких птиц, как воробьи, соловьи и другие, схожих с ними по размеру, благодаря чему есть возможность увеличить их популяцию и обеспечить более комфортное проживание во время неблагоприятных погодных условий.

Птицы различаются размерами и предпочтениями к корму. Так как птицы поменьше расходуют примерно одинаковое количество энергии для полета нужно подобрать качественный корм с достаточным для птиц калорий. Необходимо определить тип и количество корма для различных видов птиц.

Я считаю, что этот проект как никогда актуален, так как в наше время есть достаточно серьёзные проблемы с экологией и пропитанием птиц, особенно маленького размера, поэтому обязательно нужно поддерживать их популяцию и обеспечить успешную перезимовку наибольшего количества не перелетных птиц.

Изучение разнообразия птиц Битцевского лесопарка и их кормовой базы

Несмотря на довольно сильное антропогенное влияние, Битцевский лесопарк отличается достаточно большим разнообразием и высокой численностью представителей отдельных видов птиц. Наиболее часто встречающиеся [1]:

1. воробьиные (домовой воробей, полевой воробей);
2. врановые (серая ворона, ворон, сойка, грач, галка, сорока);
3. синициевые (большая синица, лазоревка, московка, буроголовая гаичка, длиннохвостая синица);
4. вьюрковые (дубонос, снегирь, зеленушка, щегол, зяблик, чечевица, чечетка, чиж);
5. дроздовые (певчий дрозд, дрозд рябинник, черный дрозд, дрозд белобровик);
6. мухоловковые (малая мухоловка, серая мухоловка, соловей, зарянка);
7. дятловые (большой пестрый дятел, средний пестрый дятел, малый пестрый дятел, белоспинный дятел, желна);
8. трясогузка (белая трясогузка, желтая трясогузка, лесной конек).

Важно понимать, что есть вредный для птиц корм, а для разных птиц есть рекомендуемые корма [2] (Таблица 1).

Таблица 1. Корма для разных видов птиц

Корм	**Особенности**	**Кто питается**
Подсолнечник (семена)	Семена должны составлять практически 70-75% всего корма (они сытны и калорийны, в них много жиров)	Синицы, дятлы, воробьи, поползни и другие зерноядные птички
Пшено	Сырая или отварная крупа (без специй и масла)	Воробьи, щеглы, голуби, зеленушки и другие зерноядные

Просо	Сухой корм (часто продается, как корм для домашних попугаев в магазинах для животных)	Воробьи, щеглы, голуби, зеленушки и другие зерноядные
Овес	Сырая или отварная крупа (без специй и масла)	Воробьи, щеглы, голуби, зеленушки и другие зерноядные
Пшеница	Сырая или отварная крупа (без специй и масла)	Воробьи, щеглы, голуби, зеленушки и другие зерноядные
Рис	Сырая или отварная крупа (без специй и масла)	Воробьи, щеглы, голуби, зеленушки и другие зерноядные
Мясо	Кусочки сырого или сушеного мяса, мелко дробленые. Без какой-либо соли и специй!	Синицы, поползни и другие виды (могут прилетать вороны, галки и сороки)
Сало	Сырое сало без соли! Его можно нанизать на нитку и подвесить	Синицы, поползни и другие виды (могут прилетать вороны, галки и сороки)
Говяжий жир или куриный	Его можно смешивать с хлебом или класть отдельно в кормушку. Жир не должен быть соленым!	Синицы, поползни и другие виды (могут прилетать вороны, галки и сороки)
Скорлупа куриного яйца	Служит хорошей кальциевой подкормкой (можно класть в кормушку кусочек натурального мела)	Для всех видов птиц

В результате анализа Таблицы 1, принято решение для разрабатываемой кормушки остановится на двух типах корма:

1. Мелкие зерноядные птицы (воробьиные, вьюрковые);
2. Синициевые, крупные зерноядные.

Выбор кормов может дорабатываться в процессе тестирования кормушки.

Анализ существующих решений

В 2022 году в США был запущен стартап Bird Buddy [3], в котором предлагается разработка умной кормушки для птиц, предназначенной для бердвотчеров - любителей наблюдать за птицами. Внешне кормушка выглядит как пластиковый скворечник с прозрачной секцией для зерна. По центру устройство оснащено камерой с Wi-Fi – она активируется, когда к кормушке прилетает птица. Камера делает снимок пернатого и отправляет на сопутствующее приложение в телефон пользователя. Стоимость кормушки – $250.

Рисунок 1. Кормушка Bird Buddy

Разработанная мною кормушка имеет принципиальное отличие в том, что будет распознавать прилетевшую птицу, а также подавать корм, который предназначен именно для нее.

3D моделирование

Модель кормушки разрабатывалась при помощи 3D моделирования. Изначально было несколько вариантов внешнего вида конструкции, но в конечном итоге я создал модель кормушки, в которой будут детали как от самых первых, так и от последующих вариантов. Модель проекта создавалась в программе Компас-3D [4] – системе проектирования, позволяющей в оперативном режиме выпускать чертежи изделий, схемы и т.д. (см. Приложение А).

Пробные варианты модели

Прототип модели «умной» кормушки имеет отличие от макета наличием жердочки и её формой. Корм проходит по трубопроводу и высыпается за счет быстрого движения сервопривода, что позволяет насыпать корм, и не рассыпая его вокруг.

В результате сборки внешний вид прототипа «умной» кормушки приобрел следующий вид:

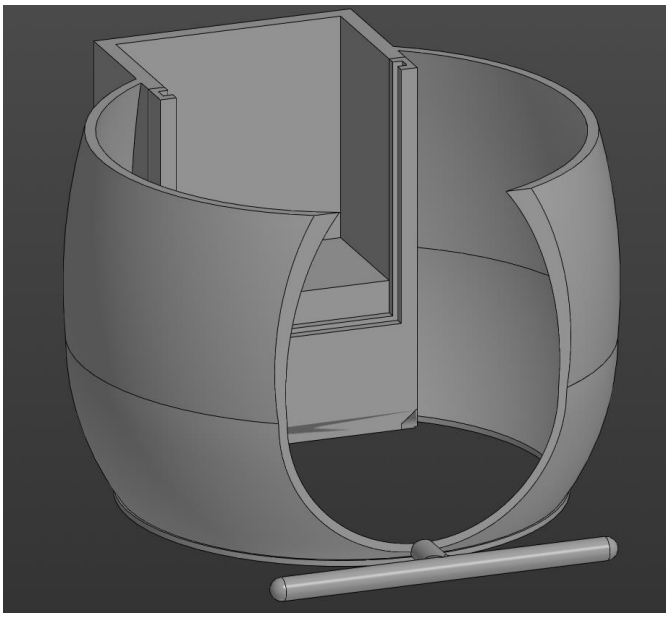


Рисунок 2. 3D модель кормушки

Рисунок 3. Пробный вариант модели кормушки

Аппаратная платформа Raspberry

Для работы кормушки был выбран Raspberry Pi 3 (рис. 4) – мощный одноплатный компьютер, который может управлять камерами, серводвигателями и различными датчиками. Преимущество Raspberry Pi перед Arduino заключается в его большей актуальности для сложных проектов, требующих большой вычислительной мощности: от ретро-игровой приставки до умного дома и небольшого сервера.

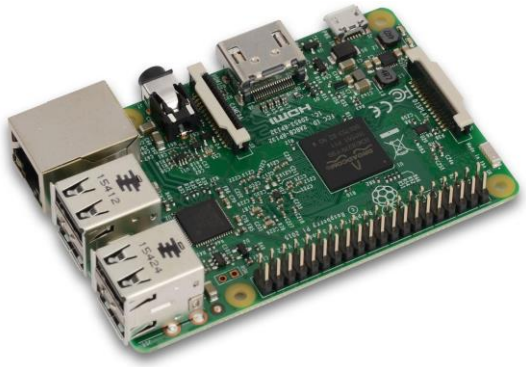


Рисунок 4. Компьютер Raspberry

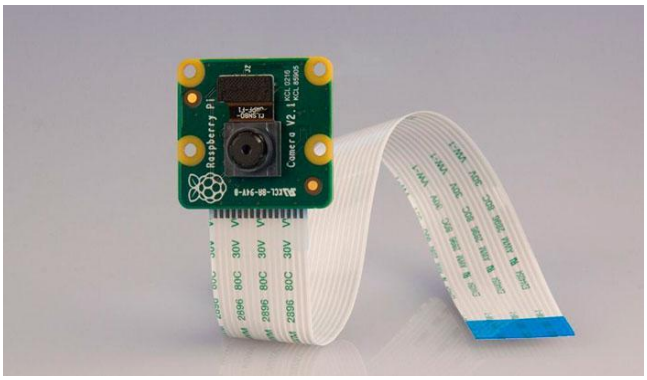
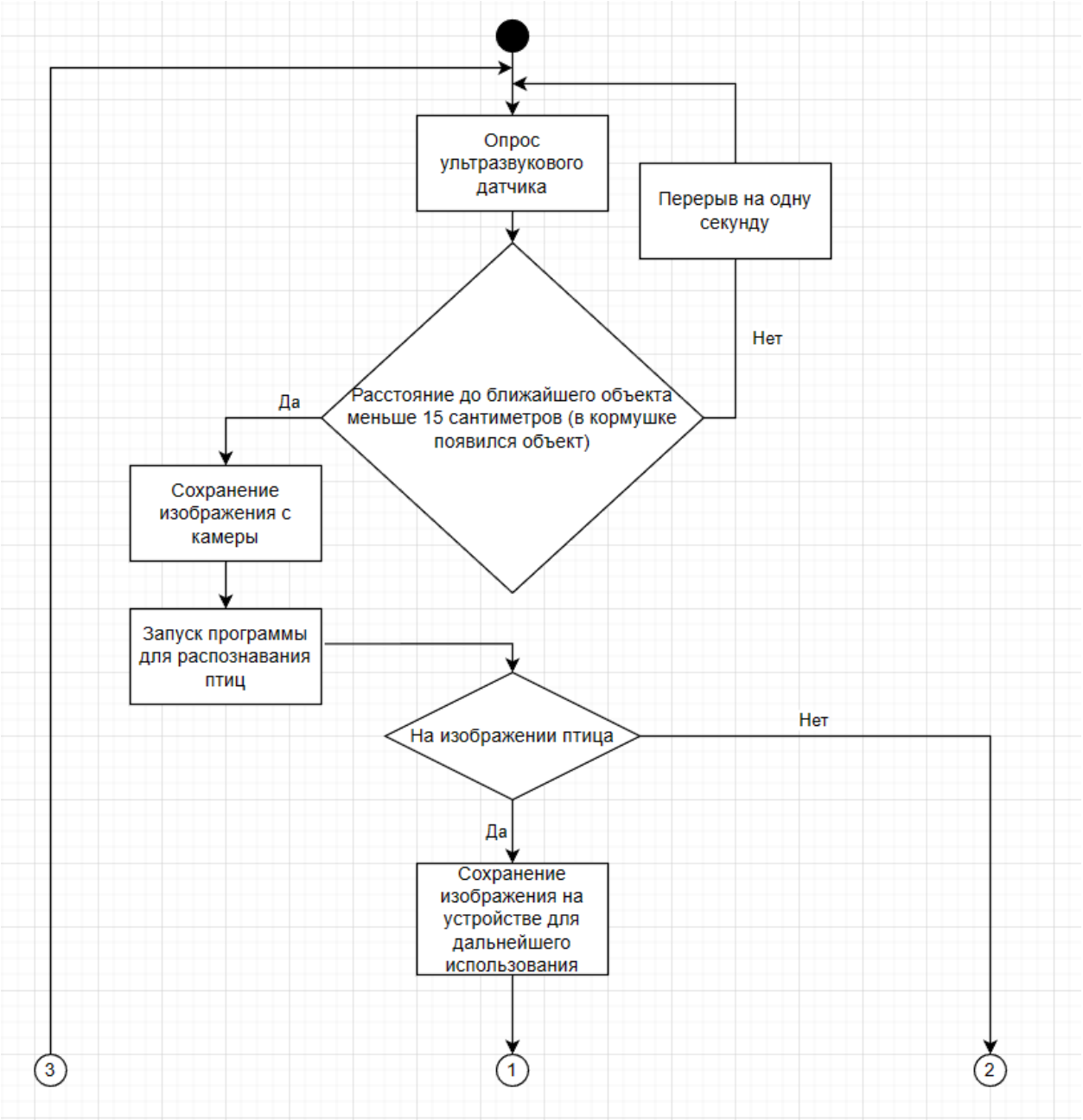


Рисунок 5. Видеокамера для Raspberry

Камера для Raspberry Pi (рис. 5) является одним из устройств ввода и представляет собой отдельный модуль на печатной плате квадратной формы, имеющей специальные отверстия для её фиксации/крепления с гибким кабелем-шлейфом, присоединяемым к разъему камеры (Последовательный интерфейс камеры, Camera Serial Interface, CSI) на плате микрокомпьютера. Я использовал эту камеру для фотографирования и дальнейшего распознавания птиц.

На рис. 6 приведен алгоритм работы умной кормушки. Программа была написана на языке python. Текст программы приведен в Приложении Б.


Опрос ультразвукового датчика
Перерыв на одну секунду
Нет
Расстояние до ближайшего объекта меньше 15 сантиметров (в кормушке появился объект)
Да
Сохранение изображения с камеры
Запуск программы для распознавания птиц
На изображении птица
Нет
Да
Сохранение изображения на устройстве для дальнейшего использования
3
1
2

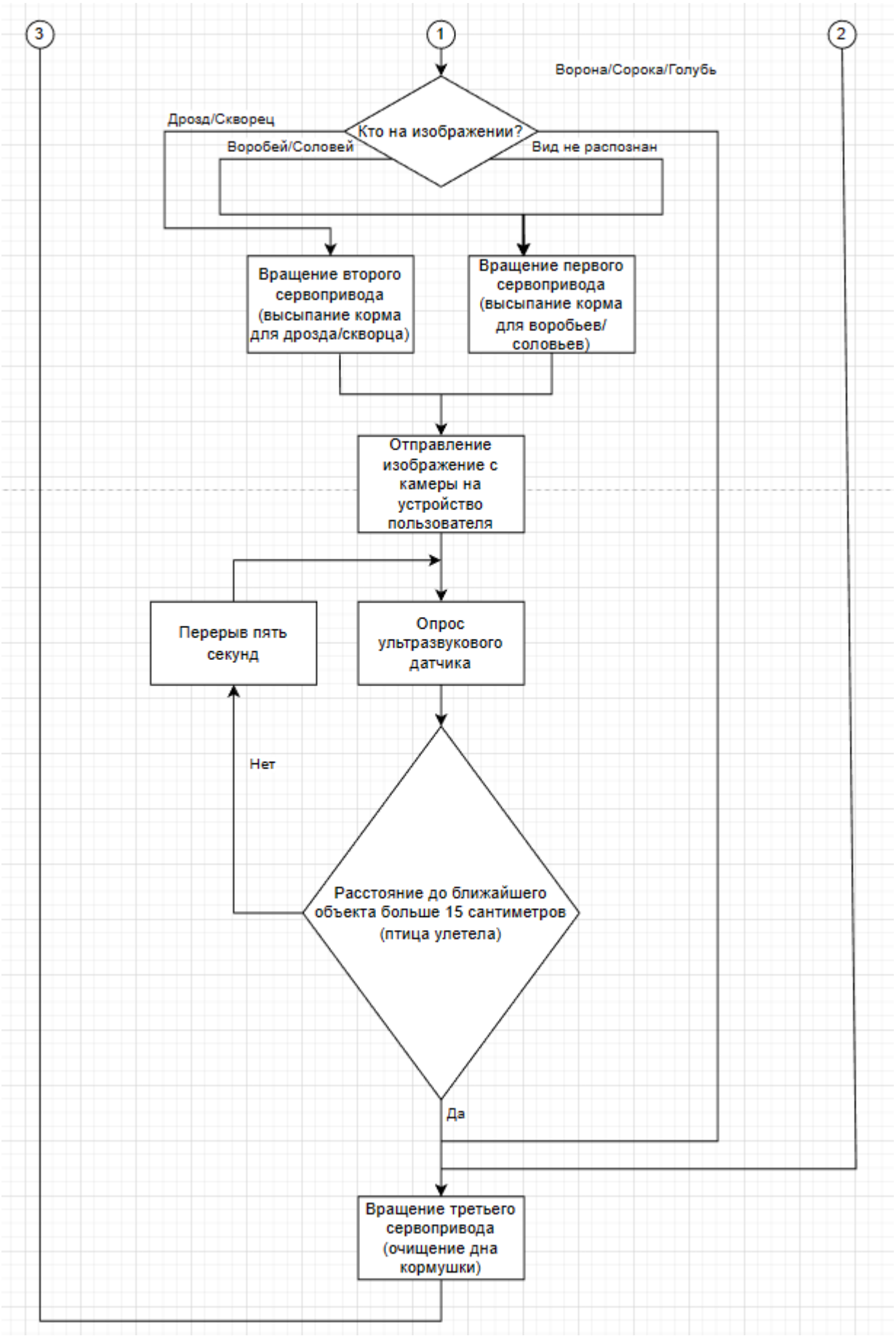


Рисунок 6. Алгоритм работы «Умной кормушки

Выбор компонентов и разработка электрической схемы

В проекте используются сервоприводы MG90S, так как они являются более дешевыми и менее потребляющими ток, нежели их аналоги.

Всего у нас используются три сервопривода:

- два сервопривода отвечают за высыпание корма и прикрепляются к их трубопроводу;
- третий сервопривод отвечает за очищение дна кормушки от собравшегося на нем мусора, образованного после кормежки птицы или попавшего внутрь корпуса случайным образом.

Принципиальная схема приведена на рис.7, а электрическая на рис.8.

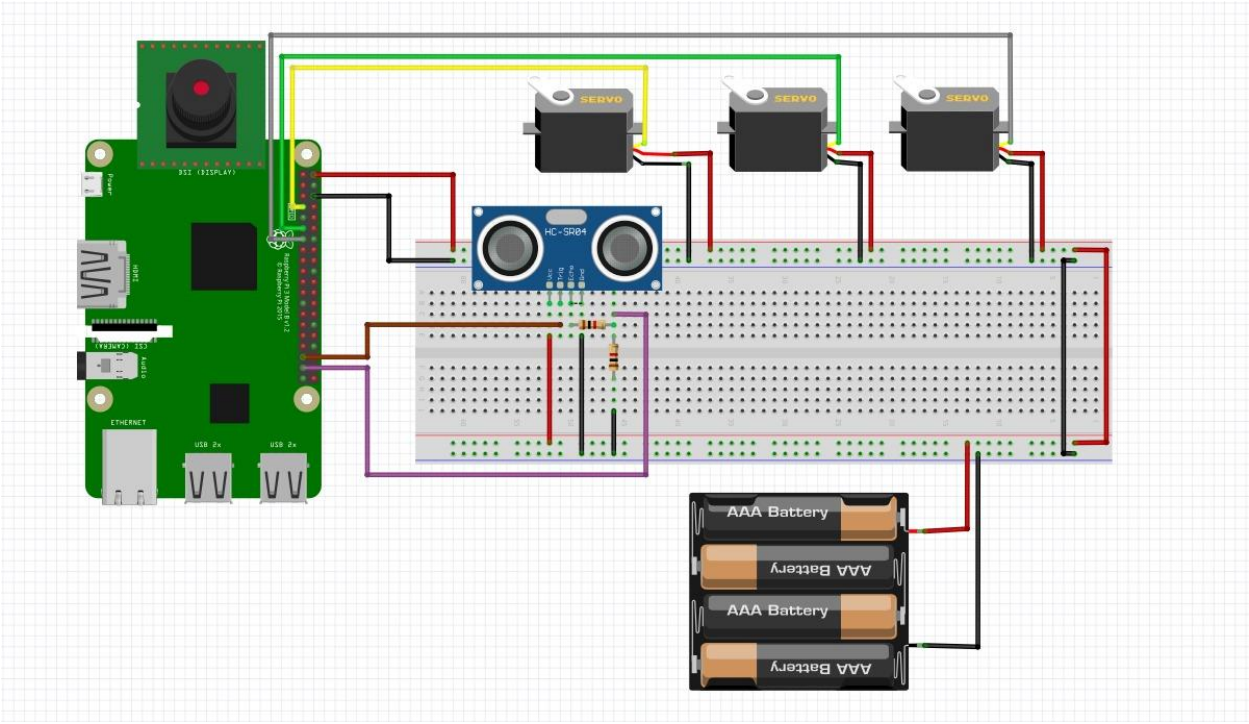


Рисунок 7. Принципиальная схема

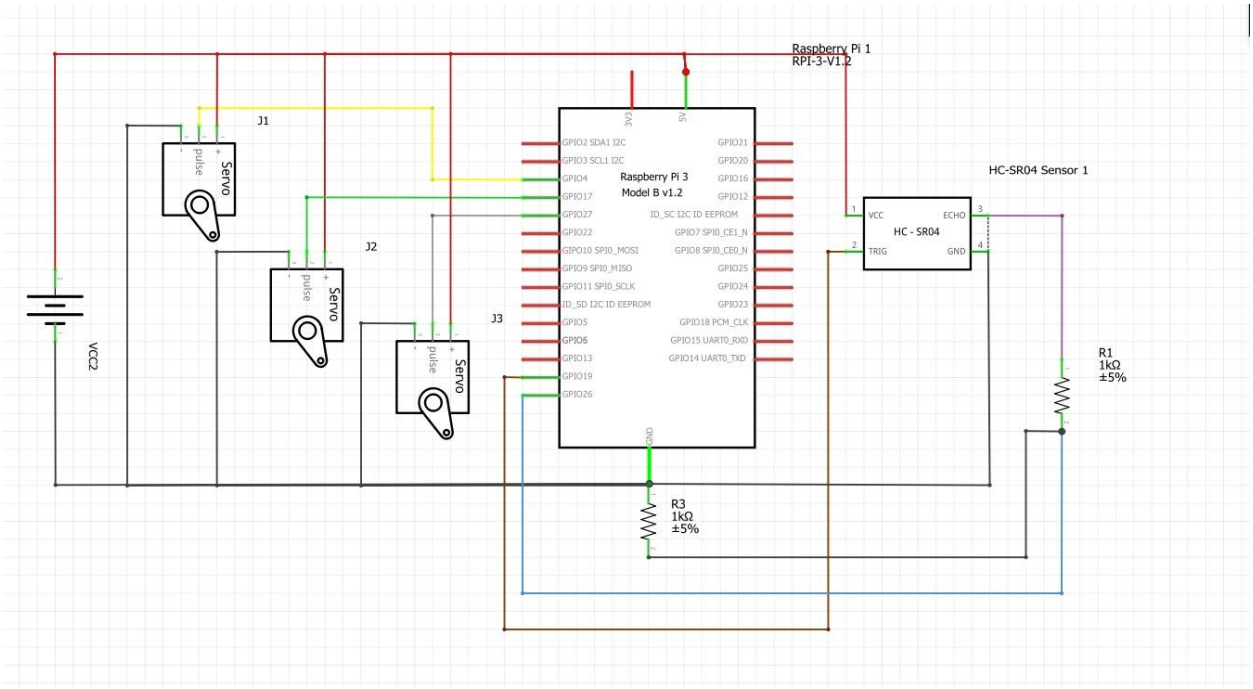


Рисунок 8. Электрическая схема

Распознавание птиц

Для распознавания фотографий птиц использовался программный интерфейс по визуальному распознаванию Microsoft Azure (рис. 9) [5]. Пользовательское визуальное распознавание Azure позволяет создавать, развертывать и улучшать пользовательские классификаторы изображений. Классификатор изображений — это служба искусственного интеллекта, которая присваивает изображениям метки контекста на основе их визуальных характеристик.

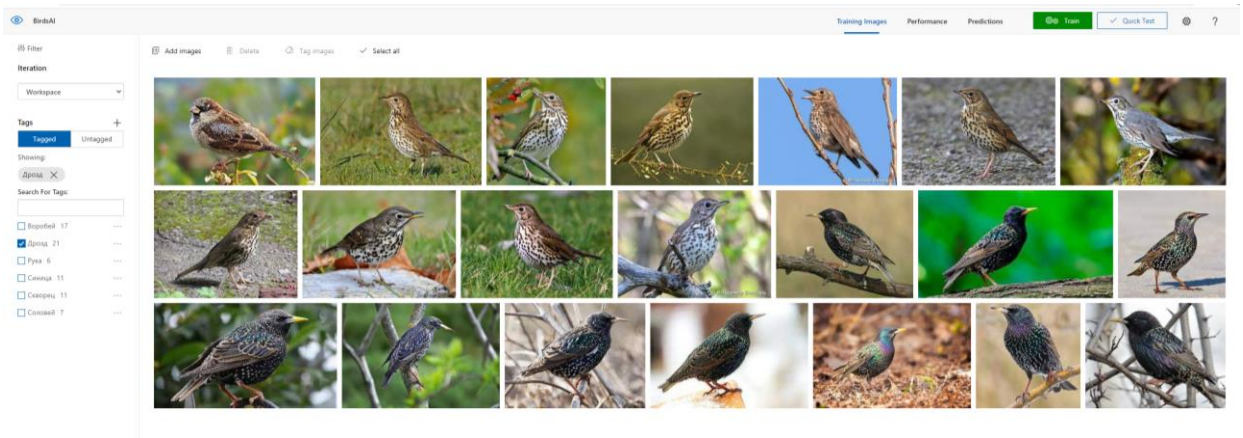


Рисунок 9: Программный интерфейс по визуальному распознаванию Microsoft Azure

Работа с данной системой включает в себя:

1. Создание проекта службы «Пользовательское визуальное распознавание».
2. Добавление тегов в проект.
3. Отправка и снабжение тегами изображений.
4. Обучение проекта.
5. Публикация текущей итерации.
6. Тестирование конечной точки прогнозирования.

Программа распознавания написана на языке Python и выполняется на компьютере Raspberry Pi 3. Результатом работы программы по визуальному распознаванию является текстовое сообщение, например, «синица», а также

подача управляющего сигнала на один из сервоприводов, отвечающих за подачу нужного корма.

Разработка телеграмм-канала

Фотографии птиц хранятся на компьютере Raspberry Pi 3. Для того чтобы была возможность их просмотра был организован телеграмм канал «Птицы» по адресу t.me/birds_794. За публикацию новых фотографий в канале отвечает телеграмм бот, который запускается в основной программе. При появлении новой фотографии осуществляется её отправка в телеграмм-канал, а также производится распознавание изображения и текстовый результат также отправляется в телеграмм-канал. Для реализации функционала телеграмм канала и бота использовалась питон библиотека pyTelegramBotAPI.

Создание готового изделия

Печать кормушки осуществлялась с помощью 3D принтера:

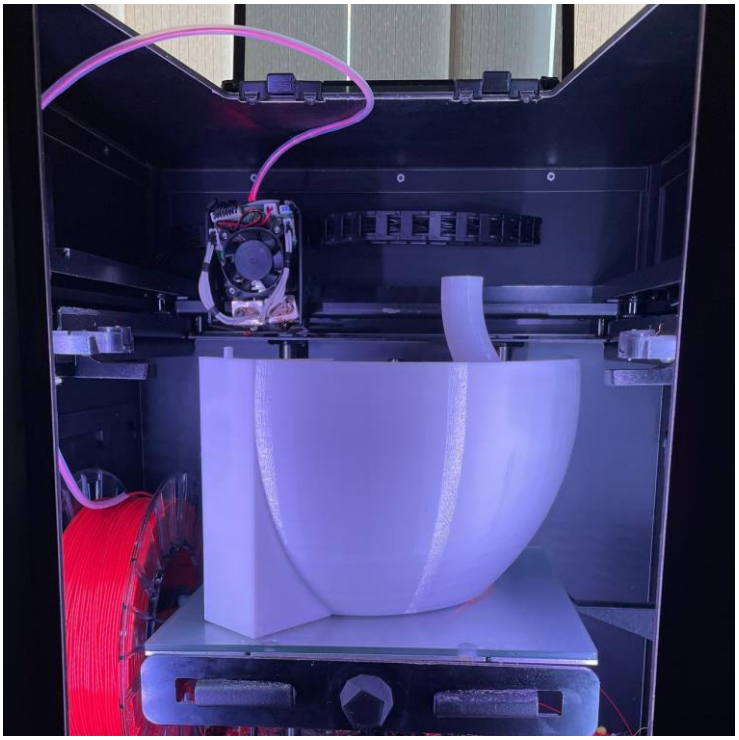


Рисунок 10. Печать кормушки на 3D принтере

Технологическая карта

Основные технологические операции представлены в технологической карте (табл. 2). Общее время работы составило 159 часов.

Таблица 2. Технологическая карта

Описание технологических операция	**Используемое оборудование**	**Время (часы)**
Моделирование в САПР-программах	Компас-3D	36
Сборка электрической схемы	Fritzing	1
Программирование системы распознавания птиц	PyCharm, Raspberry Camera	28
Настройка ультразвукового дальномера	Raspberry Pi 3, ультразвуковой дальномер HC-SR04	2
Настройка сервоприводов	Raspberry Pi 3, Servo MG90S	2
Печать модели	3D-принтер	58
Сборка кормушки	Клей, струбцины	2
Тестирование		30
Итого		**159**

Итоги и перспективы

На данный момент создана «умная кормушка», выполняющая такие функции, как обнаружение объекта, распознавание птицы на изображении, высыпание двух видов корма для птиц в зависимости от распознанной на изображении птицы, отправление изображения с камеры на устройство и очищение дна кормушки. В будущем планируется доработать систему распознавания птиц, подключить системы обогрева и освещения кормушки и разработать приложение для данного проекта, чтобы в нем пользователь в прямом эфире мог наблюдать за поведением пернатого друга.

В Соединенных Штатах Америки существует аналог данного проекта под названием «Bird Buddy». Кормушка с камерой на базе искусственного интеллекта уведомляет пользователя о птицах-посетителях, делает их фотографии и объединяет их в коллекцию на устройстве владельца. Заряда хватает на 10-20 дней при теплой погоде, на морозе она работает еще меньше, после чего необходима перезарядка дома. Также кормушка хранит один вид корма, который может для одних птиц быть полезным, а для других - вредным. Моя же кормушка использует информацию о том, какая птица прилетела, чтобы высыпать ей предназначенный для нее корм, и в будущем время ее работоспособности будет составлять около двух-трех недель. К тому же, Bird Buddy в США по нынешнему курсу стоит около двадцати тысяч рублей, а в России она продается за тридцать тысяч. У моего же проекта себестоимость составляет около 17 тысяч рублей - с учетом Raspberry, сервоприводов и т.д. - при этом для данной кормушки можно использовать более дешевые платы — например, Micropython pyboard — и тогда себестоимость проекта будет составлять около 5 тысяч рублей. Поэтому мой проект в России будет более выгодным и более полезным в сравнении с указанным аналогом.

Хотелось бы отметить, что также нужно предусмотреть местонахождение кормушки, так как я планирую следить за питанием птиц, поэтому потребуется выбрать более тихий и спокойный участок. Это

требуется для того, чтобы прохожие не могли мешать подготовленный корм с пищей, которая может быть вредна для птиц. Для этого потребуется договориться с управляющими парка или орнитологами о том, чтобы повесить у кормушек таблички, запрещающие подсыпать иные корма, а также о контроле и поддержании нужного количества корма.

Список литературы

1. Птицы Битцевского леса. [Электронный ресурс] // [сайт]. — URL: https://bitsa-forest.livejournal.com/100050.html (дата обращения: 07.11.2022).
2. Чем можно и чем нельзя кормить птиц зимой в кормушке [Электронный ресурс] // [сайт]. — URL: https://eggincubator.ru/ptica/mozhno-li-kormit-sinic-grechkoj.html (дата обращения: 15.12.2022).
3. Умная кормушка Bird Buddy кормушке [Электронный ресурс] // [сайт]. — URL: https://mybirdbuddy.com/ (дата обращения: 20.12.2022).
4. Система трехмерного моделирования Компас 3D. [Электронный ресурс] // [сайт]. — URL: https://kompas.ru/ (дата обращения: 15.12.2022).
5. Raspberry Pi. Официальное Руководство Для Начинающих [Электронный ресурс] // [сайт]. — URL: https://www.raspberrypi.com/documentation/computers/getting-started.html. (дата обращения: 18.01.2023).
6. Custom Vision. [Электронный ресурс] // [сайт]. — URL: https://www.customvision.ai/ (дата обращения: 12.02.2023).Документация по Пользовательскому визуальному распознаванию Microsoft Azure [Электронный ресурс] // [сайт]. — URL: https://learn.microsoft.com/ru-ru/azure/cognitive-services/custom-vision-service/ (дата обращения: 10.02.2023).

Приложение А. Чертежи деталей модели

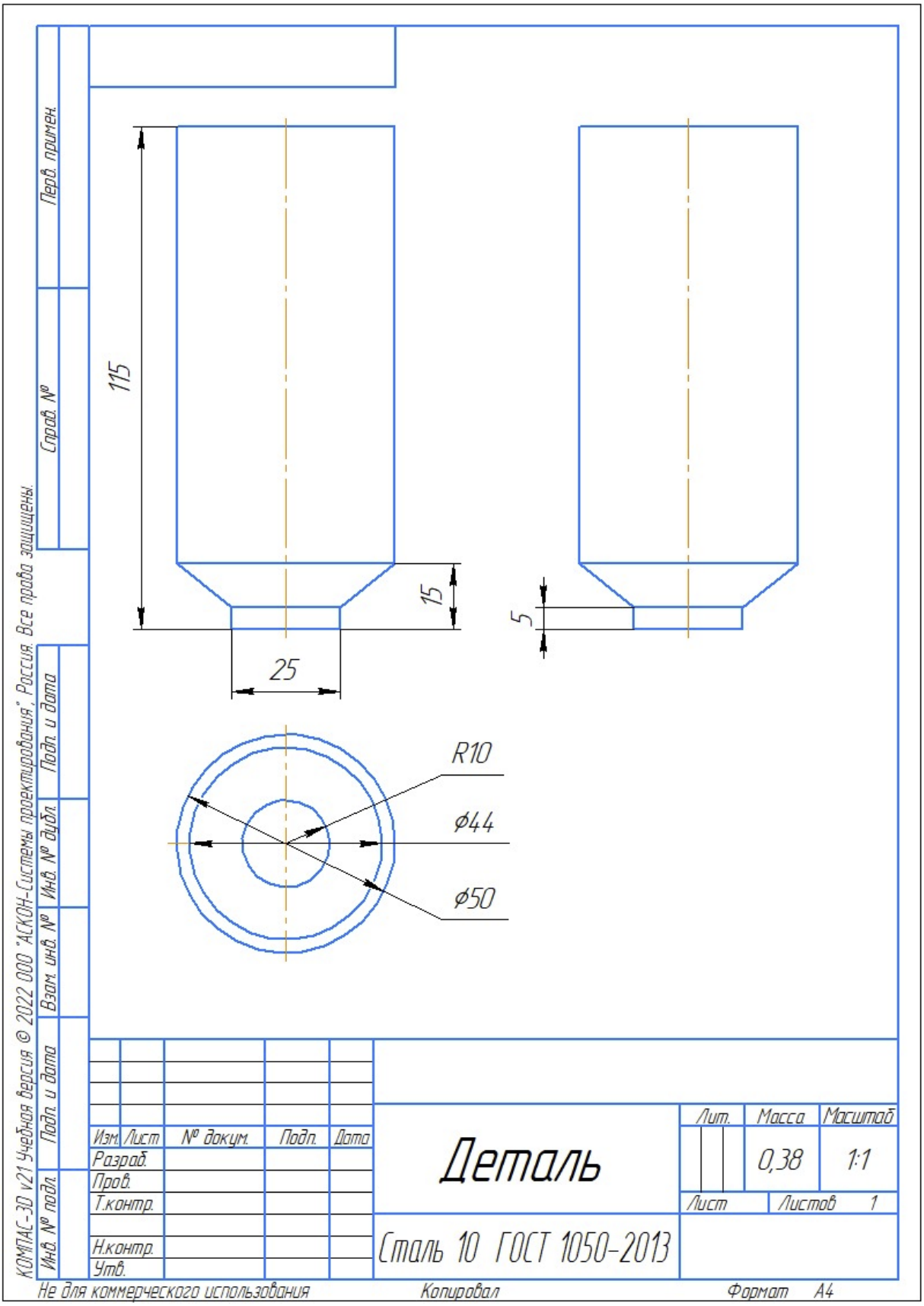


Рисунок 11. Чертеж бака для корма

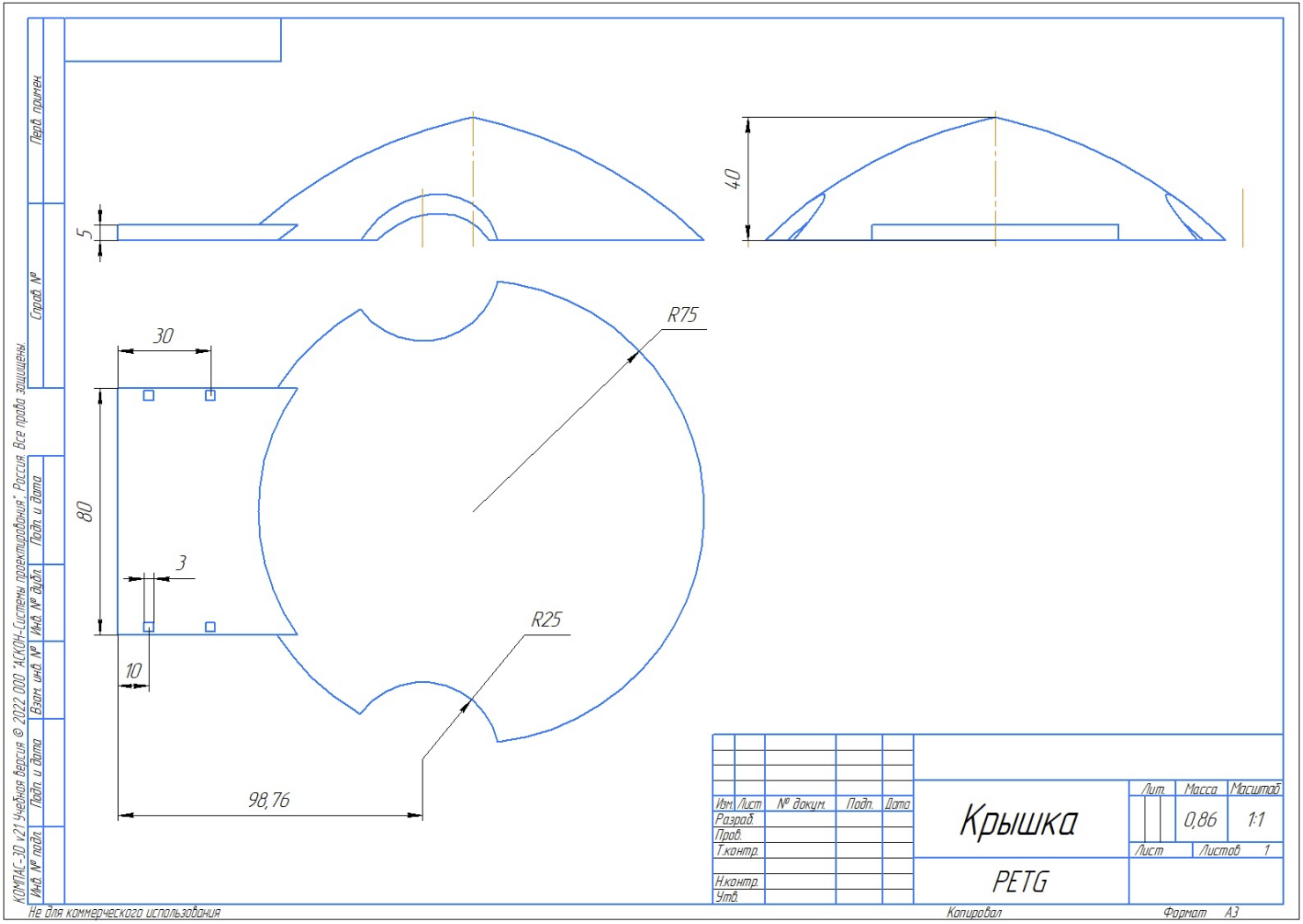


Рисунок 12. Чертеж крышки кормушки

Приложение Б. Листинг программы

```
import RPi.GPIO as IO
# подключение библиотеки для работы с контактами ввода/вывода
import time
# подключение библиотеки для работы с задержками
IO.setwarnings(False)
# отключаем показ любых предупреждений
IO.setmode (IO.BCM)
# мы будем программировать контакты GPIO по их функциональным номерам (BCM), то есть мы будем обращаться к PIN29 как ‘GPIO5’
IO.setup(17,IO.OUT)
# инициализируем GPIO17 в качестве цифрового выхода
p = IO.PWM(17,50)
# инициализируем GPIO17 как контакт для формирования ШИМ сигнала с частотой 50 Гц
IO.setup(4,IO.OUT)
# инициализируем GPIO4 в качестве цифрового выхода
q = IO.PWM(4,50)
# инициализируем GPIO4 как контакт для формирования ШИМ сигнала с частотой 50 Гц
p.start(7.5)
# генерируем ШИМ сигнал с коэффициентом заполнения 7.5%
q.start(7.5)
# генерируем ШИМ сигнал с коэффициентом заполнения 7.5%
while 1:
# бесконечный цикл
        p.ChangeDutyCycle(6)
# изменяем коэффициент заполнения чтобы повернуть сервомотор в положение 90º
        q.ChangeDutyCycle(6)
        time.sleep(3)
        p.ChangeDutyCycle(1)
# изменяем коэффициент заполнения чтобы повернуть сервомотор в положение 0º
        q.ChangeDutyCycle(1)
        time.sleep(5)
# задержка на 1 секунду
```

```
import RPi.GPIO as GPIO
 import time
 import cv2
 f = False
 GPIO.setwarnings(False)
 GPIO.setmode(GPIO.BCM)
 TRIGGER=19
 ECHO=26
 GPIO.setup(TRIGGER, GPIO.OUT)
 GPIO.setup(ECHO, GPIO.IN)
 def ultra():
    global f
    GPIO.output(TRIGGER, GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(TRIGGER, GPIO.LOW)
    while GPIO.input(ECHO) == 0:
       start = time.time()
    while GPIO.input(ECHO) == 1:
       end = time.time()
    timepassed = end - start
    distance = round(timepassed * 17150, 2)
    print("The distance from object is ",distance,"cm")
    if distance < 15 and not f:
       cap = cv2.VideoCapture(0);
       _, frame = cap.read()
       cv2.imwrite("/home/pi/Documents/Birds/Bird.png", frame)
       f = True
    elif distance >= 15 and f:
       f = False
 while True:
    ultra()
    time.sleep(1)
```

```
        import cv2
# импорт модуля cv2
        import telebot
# импорт модуля  для телеграмм канала
        bot =
telebot.TeleBot('5866226591:AAFHoLsDmMU8siud04rxskTkZCPKllgydcY')
        CHANNEL_NAME = '@birds_794'
# Адрес телеграм-канала
        img = open(path, 'rb')
# path — путь к изображению
        bot.send_photo(CHANNEL_NAME, img)


         import tensorflow as tf
         import os
         from PIL import Image
         import numpy as np
         import cv2


         def convert_to_opencv(image):
            # RGB -> BGR conversion is performed as well.
            image = image.convert('RGB')
            r,g,b = np.array(image).T
            opencv_image = np.array([b,g,r]).transpose()
            return opencv_image

         def crop_center(img,cropx,cropy):
            h, w = img.shape[:2]
            startx = w//2-(cropx//2)
            starty = h//2-(cropy//2)
            return img[starty:starty+cropy, startx:startx+cropx]

         def resize_down_to_1600_max_dim(image):
            h, w = image.shape[:2]
            if (h < 1600 and w < 1600):
               return image

            new_size = (1600 * w // h, 1600) if (h > w) else (1600, 1600 * h // w)
            return cv2.resize(image, new_size, interpolation = cv2.INTER_LINEAR)

         def resize_to_256_square(image):
            h, w = image.shape[:2]
            return cv2.resize(image, (256, 256), interpolation = cv2.INTER_LINEAR)
```

```
    def update_orientation(image):
        exif_orientation_tag = 0x0112
        if hasattr(image, '_getexif'):
            exif = image._getexif()
            if (exif != None and exif_orientation_tag in exif):
                orientation = exif.get(exif_orientation_tag, 1)
                # orientation is 1 based, shift to zero based and flip/transpose based on
0-based values
    orientation -= 1
    if orientation >= 4:
                image = image.transpose(Image.TRANSPOSE)
    if orientation == 2 or orientation == 3 or orientation == 6 or orientation == 7:
                image = image.transpose(Image.FLIP_TOP_BOTTOM)
    if orientation == 1 or orientation == 2 or orientation == 5 or orientation == 6:
                image = image.transpose(Image.FLIP_LEFT_RIGHT)
        return image

    graph_def = tf.compat.v1.GraphDef()
    labels = []

    # These are set to the default names from exported models, update as needed.
    filename = "model.pb"
    labels_filename = "labels.txt"

    # Import the TF graph
    with tf.io.gfile.GFile(filename, 'rb') as f:
        graph_def.ParseFromString(f.read())
        tf.import_graph_def(graph_def, name='')

    # Create a list of labels.
    with open(labels_filename, 'rt') as lf:
        for l in lf:
            labels.append(l.strip())

    # Load from a file
    imageFile = "test.jpg" # тут ссылка на фото
    image = Image.open(imageFile)

    # Update orientation based on EXIF tags, if the file has orientation info.
    image = update_orientation(image)

    # Convert to OpenCV format
    image = convert_to_opencv(image)
```

```
        # If the image has either w or h greater than 1600 we resize it down respecting
        # aspect ratio such that the largest dimension is 1600
        image = resize_down_to_1600_max_dim(image)

        # We next get the largest center square
        h, w = image.shape[:2]
        min_dim = min(w,h)
        max_square_image = crop_center(image, min_dim, min_dim)

        # Resize that square down to 256x256
        augmented_image = resize_to_256_square(max_square_image)

        # Get the input size of the model
        with tf.compat.v1.Session() as sess:
            input_tensor_shape                                                    =
sess.graph.get_tensor_by_name('Placeholder:0').shape.as_list()
        network_input_size = input_tensor_shape[1]

        # Crop the center for the specified network_input_Size
        augmented_image = crop_center(augmented_image, network_input_size,
network_input_size)
        # These names are part of the model and cannot be changed.
        output_layer = 'loss:0'
        input_node = 'Placeholder:0'
        with tf.compat.v1.Session() as sess:
            try:
                prob_tensor = sess.graph.get_tensor_by_name(output_layer)
                predictions = sess.run(prob_tensor, {input_node: [augmented_image] })
            except KeyError:
                print ("Couldn't find classification output layer: " + output_layer + ".")
                print ("Verify this a model exported from an Object Detection project.")
                exit(-1)
# Print the highest probability label
            highest_probability_index = np.argmax(predictions)
            print('Classified as: ' + labels[highest_probability_index])
            print()
  # Or you can print out all of the results mapping labels to probabilities.
            label_index = 0
            for p in predictions:
                truncated_probablity = np.float64(np.round(p,8))
                #print (labels[label_index], truncated_probablity)
                label_index += 1
```

Приложение В. Фотографии изделия