**Автономная некоммерческая общеобразовательная организация "Физтех-лицей"
(АНОО «Физтех-лицей» им. П.Л. Капицы)**

XX научно-практическая конференция

«Старт в инновации»

Сборка и программирование беспилотного автомобиля

Выполнили:
Жумаева Виктория
Коган Владислав
Пискунов Всеволод
10 класс
Руководитель:
Обухан Вячеслав Геннадьевич

Московская область, г. Долгопрудный
2021 г.

Оглавление

Введение

Наша команда задумалась над идеей создания беспилотной машины, так как мы видим глобальную тенденцию перехода к транспорту без водителя. Беспилотники обладают рядом преимуществ по сравнению с обычными автомобилями, например они безопасны и способствуют снижению загруженности на дорогах. В нашем проекте мы создали модель такого беспилотника, который умеет ориентироваться в пространстве без использования человеческих рук.

Цели:

- Собрать беспилотный автомобиль
- Написать программу для автомобиля на Python
- Запрограммировать автомобиль

Задачи:

- Освоить азы программирования на языке Arduino IDE
- Разобраться в работе электронных схем и их компонентов.
- Изучить машинное зрение и нейронные сети

Методы исследования:

Моделирование, анализ, эксперимент, наблюдение.

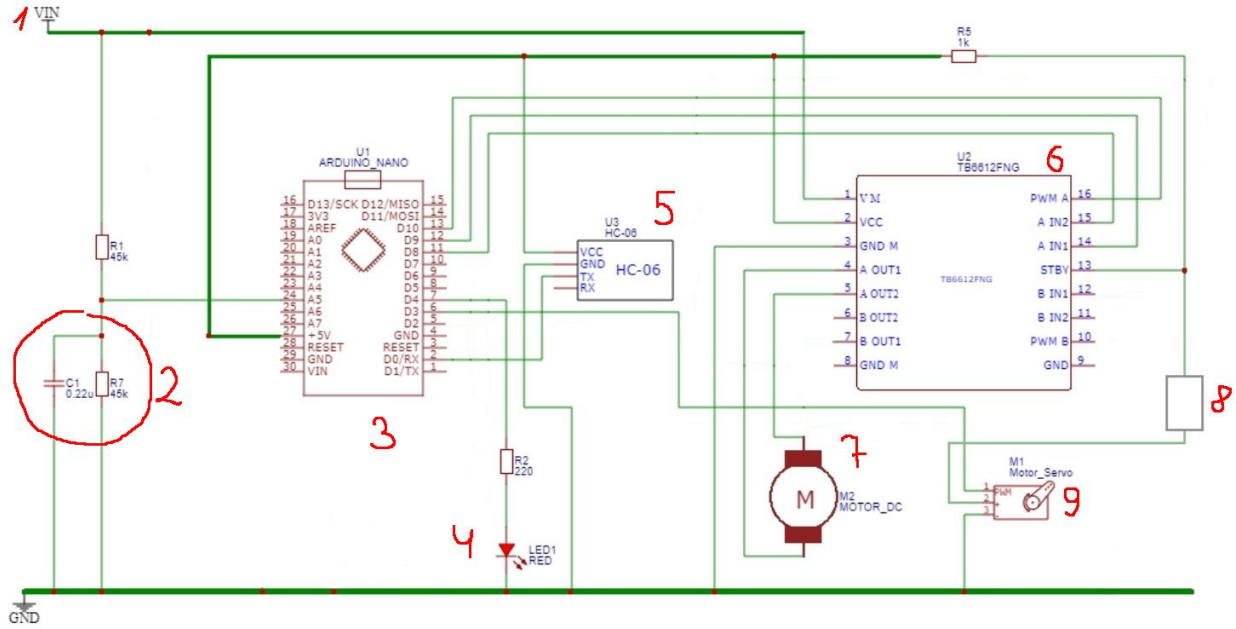
Главный результат:

Автомобиль различает дорожные знаки и объезжает видимые препятствия.

1. Сборка автомобиля и описание деталей

Корпус автомобиля, мотор мы взяли от радиоуправляемой машины, плату взяли из проекта Теннис. Провода и камеру позаимствовали из ФОК 205. Машинку собирали вместе с научным руководителем.

Электрическая схема логической составляющей архитектуры машинки и моторов:



1. Основной аккумулятор отвечает за подачу напряжения на плату Arduino и H-мост. Напряжение подаваемое основным аккумулятором колеблется в пределах от 7 до 9 В.

2. Эта часть цепи компенсирует скачки напряжения, а R1 уменьшает напряжение, подаваемое на плату Arduino Nano.

3. Плата Arduino получает сигнал с Bluetooth-приемника и отправляет сигнал с соответствующими действиями на сервомотор и сигнал на H-мост. Также она передает напряжение 5 В на H-мост.

4. Красный светодиод загорается, когда напряжение, подаваемое на плату Arduino, меньше допустимого значения.

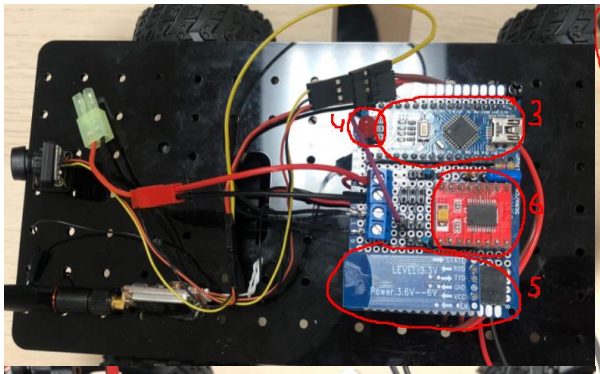
5. Bluetooth-приемник передает сигнал, поступающий с компьютера в Arduino.

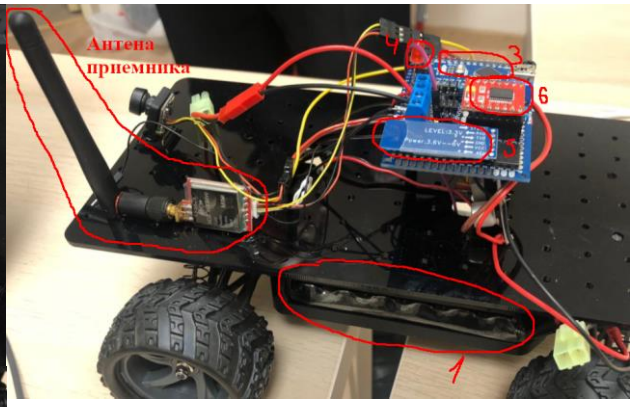
6. H-мост передает на мотор сигнал, содержащий указания по скорости и направлению езды.

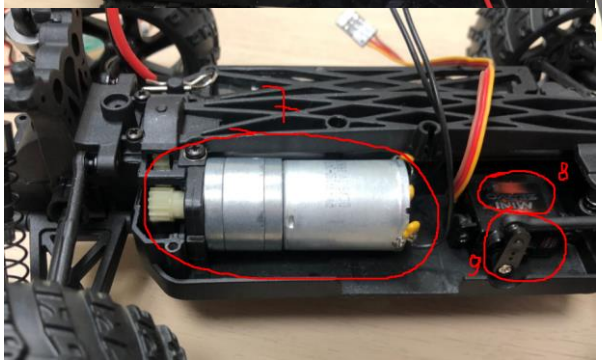
7. Мотор, питается от H-моста.

8. Второй, более маленький аккумулятор отвечает за подачу напряжения на сервомотор (5-6 В).

9. Сервомотор, отвечает за положение колес автомобиля во время езды.



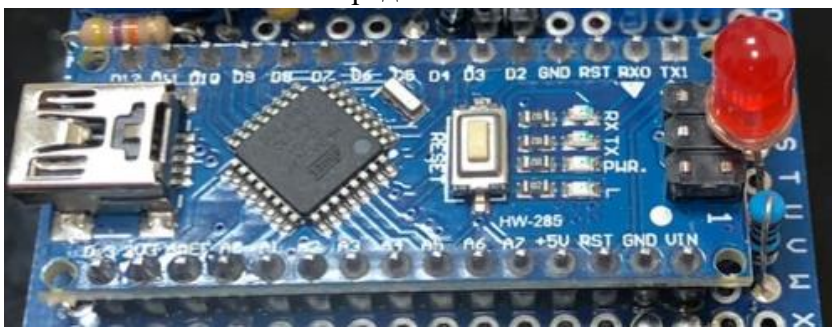




2. Программирование автомобиля

2.1. Краткое содержание

Машинка должна распознавать дорожные знаки и что-то делать, в соответствии с тем, какой знак она распознала. Для этого на машинке есть камера, изображение с камеры с камеры передается по Bluetooth на компьютер, там оно обрабатывается при помощи компьютерного зрения и нейронной сети, далее информация о том, какой знак видит автомобиль по Bluetooth передается на Arduino Nano.



В Arduino записана специальная программа, которая в соответствии с тем, какая информация на нее поступает, отправляет определенный сигнал на мотор и сервомотор машинки.

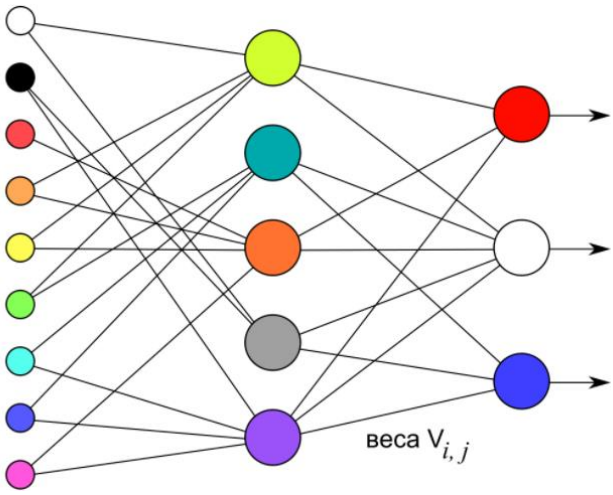
2.2. Детектирование знака с помощью компьютерного зрения

Чтобы распознать знак, машинка сначала должна понять, что такое знак. Какие отличительный черты у знака? Знаки в нашей подборке имеют либо синий, либо красный цвет. Знаки на изображении можно детектировать по цвету на Python, используя библиотеку OpenCV. Для этого считываем кадр с камеры, переводим картинку из цветовой модели BGR (Blue, Green, Red) в HSV (Hue, Saturation, Value — тон, насыщенность,

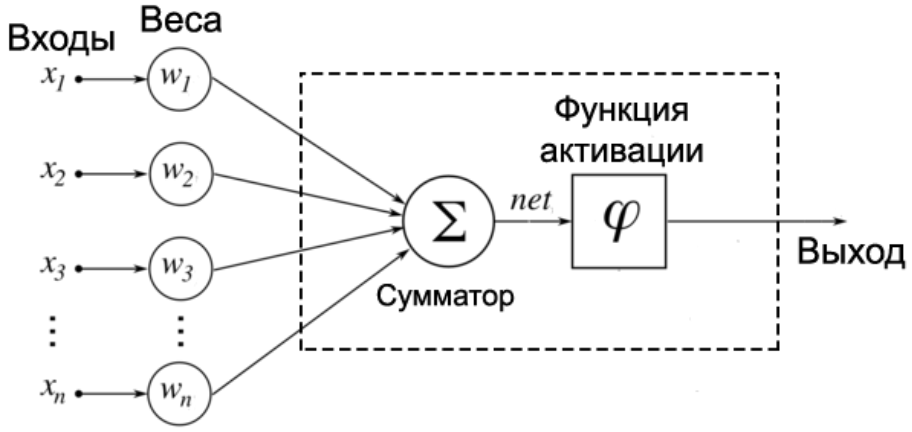
яркость), размываем картинку, чтобы убрать рябь, и бинаризуем изображение. Бинаризация - это преобразования изображения в черно-белый формат, причем мы можем настраивать пороги бинаризации, то есть можно написать, при каких значениях тона, насыщенности и яркости пикселя он будет белым. Дальше применяем операции erode (убирает мелкие белые пиксели) и dilate (убирает мелкие черные пиксели). В итоге остаются контуры крупных объектов. В OpenCV есть функции, которые могут сделать массив из контуров и отсортировать контуры по убыванию. Нас интересует самый большой контур. Записываем его координаты и вырезаем из изображения этот контур.

2.3. Распознавание дорожного знака с помощью сверточной нейронной сети

Что такое нейронная сеть? Как она работает? Преимущество нейронной сети перед компьютерным зрением в том, что для того, чтобы написать программу распознавания на компьютерном зрении, надо выделить отличительные признаки объекта, который хотим распознать, сформулировать эти признаки на языке машины. Нейронная сеть же сама находит признаки и делает это в разы быстрее человека. Есть много видов нейронных сетей. Для распознавания образов чаще всего используют сверточную нейронную сеть. Схематично нейронную сеть можно представить так:



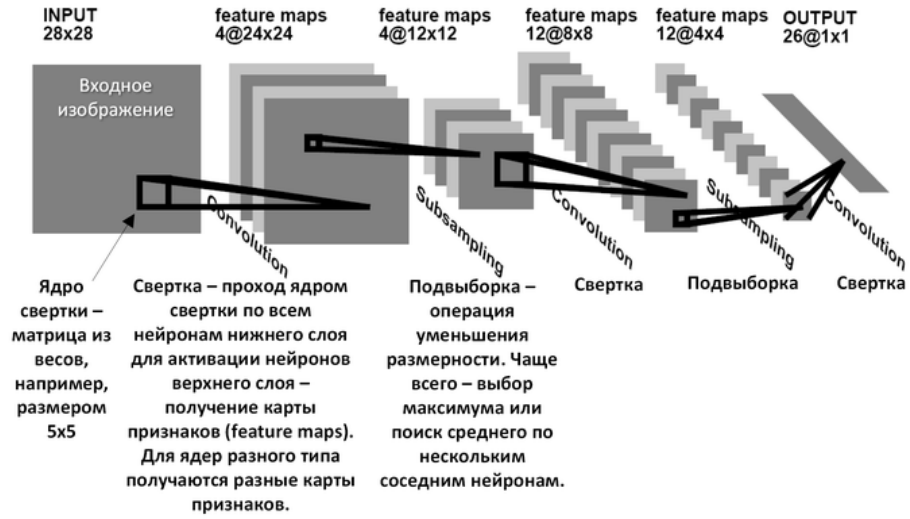
Это один слой нейронной сети:

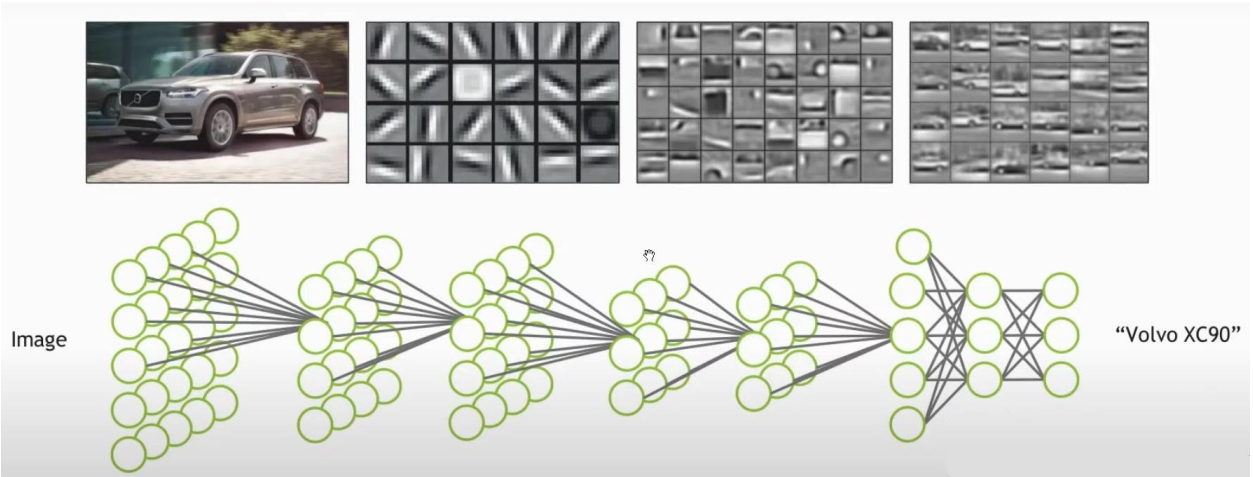


Преимущества сверточной нейронной сети:

1) Умеет извлекать из изображения признаки;
2) Учитывает пространственное расположение пикселей;
3) Меньшее количество весовых коэффициентов, которые подбирает сеть.

Принцип работы сверточной нейросети:





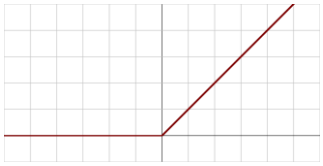
Мы создали свою нейронную сеть для распознавания дорожных знаков.

Для этого мы использовали приложение Google Colaboratory, так как там есть все нужные библиотеки. Писали на Python. Основные библиотеки, которые нам понадобились: Tensorflow, Numpy, Pickle, Os, Random и OpenCV.

1) Нужны данные, на которых будет обучаться нейросеть (датасет). Датасет мы взяли с сайта sim.newgen.education. В папке с данными есть 3 папки: training, val и test. В каждой из них есть 8 папок с фотографиями детектированных знаков и одна папка с фотографиями, на которых нет знаков. Эти папки мы загрузили на Google диск и в программе указали путь к файлам. Дальше мы сделали массив с изображениями, у которых сделали одинаковый размер (изображения следует перемешать, используя функцию random), и массив с метками, который показывает, к какому классу принадлежит фотография. Массивы сохранили на диске.

2) Надо написать структуру сверточной нейронной сети.

Модель нейронной сети - Sequential (последовательная), то есть слои идут один за другим. Размер входных данных RGB картинка 32 на 32. Дальше добавляем в модель сверточный слой, который имеет 8 карт признаков и размер ядра свертки на каждой карте 5 на 5. Размер карты признаков сохраняем. Функция активации relu.



Relu

Нормализуем входной слой. Дальше идет слой подвыборки, который выбирает из картинки признаки, которые имеют наибольшее значение. Потом идут 2 слоя свертки, слой подвыборки, 2 слоя свертки, слой подвыборки. Следующий слой Flatten, который переводит двумерную картинку в одномерный вектор, чтобы передать эти данные следующему слою - Dense (полносвязный слой, работает с одномерными векторами). Также после функции активации и нормализации идет функция Dropout, которая отключает часть нейронов, чтобы следующие нейроны не полагались на предыдущие. Повторяем слой Dense с Dropout и прописываем последний выходной слой Dense, в котором 9 нейронов. Это 9 классов изображений: a_unevenness, no_drive, no_entry, parking, pedestrian, road_works, stop, way_out, none. Функция активации softmax.

3) Дальше идет обучение нейронной сети. Загружаем массивы с картинками и с метками. Разбиваем данные на тренировочные и тестовые (15% от тренировочных) (на тренировочных нейросеть обучается, на тестовых тестируется). Надо узнать, какие веса у каждого класса. Для этого есть специальная функция. Из-за того, что изображений каждого знака разное количество, нейросеть будет распознавать одни знаки лучше, другие хуже. Чтобы это исправить, можно прописать каждому классу веса вручную. Как будет проходить обучение? Мы сделали 30 эпох (одна эпоха - один цикл обучения). Начальная скорость обучения = 1e-3, batch size = 64 (это то, сколько картинок подается за раз). Алгоритм оптимизации Adam. Также можно разнообразить датасет функцией ImageDataGenerator, которая может сдвигать, поворачивать, приближать, отдалять исходное изображение. Еще нужно прописать чекпоинт, который сохраняет лучшую модель. Казалось бы, почему бы не взять модель, которая прошла все 30 эпох, она ведь больше обучилась? Нейронные сети могут переобучиться, то есть просто запоминать картинки, которые ей даются во время обучения, и тогда она не будет искать новых признаков, и если ей дать совершенно новое изображение, нейросеть его не классифицирует.

4) Последний блок кода тестирует нейронную сеть на данных, которые она еще не видела, считает ее точность и сохраняет сеть на диск.

5) Теперь можно скачать нейросеть на компьютер и поместить ее в папку с программой. Загружаем модель и подаем ей на вход изображение детектированного знака. Она его распознает и посылает определенный сигнал в Arduino. Так же чтобы машинка не реагировала на знак слишком рано, есть условие на размер исходного кадра.

2.4. Обработка сигнала в Arduino

После определения вида знака, программа отправляет одну из 5 латинских букв на плату Arduino, с помощью функции ArduinoSerial.write(str.encode(“буква”)).

Каждая буква соответствует единственному знаку.

“F” - Forward - машина должна двигаться вперед

“B” - Back - машина должна двигаться назад

“S” - Stop - машина должна остановиться

“L” - Left - машина должна повернуть налево

“R” - Right - машина должна повернуть направо

Далее, Arduino получает эту букву и записывает ее в переменную data, используя функцию Serial.read().

В коде Arduino идет проверка переменной data на равенство определенной букве. Например, если data равняется “F” (машинка должна ехать вперед) выполняется следующий блок кода:

```
if (data == ‘F’) {

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, HIGH);
```

Две данные функции переключают мотор в режим движения вперед;

```
analogWrite(pwmPin,600);
```

Данная функция меняет скорость мотора на 600 единиц (Скорость может изменяться от 0 до 1023 единиц)

```
}
```

Также в программе использовалась функция myservo.write(“угол”) которая отвечала за поворот руля на определенный угол.

Данный блок кода выполняется с задержкой в 100мс.

Чтобы отслеживать уровень зарядки аккумулятора, в программе присутствует переменная volt. Блок кода с проверкой букв описанный ранее выполняется только в том случае, если volt>650 - условие означает достаточный заряд аккумулятора для работы машинки.

Если камере не удается детектировать знак, машинка продолжает свое движение по прямой линии.

Плата Arduino UNO отправляет сигнал компонентам машинки, соединительным проводами. Более детальное описание взаимосвязи машинки и компьютера представлено ниже.

Заключение

Мы собрали и запрограммировали беспилотный автомобиль. В ходе нашей деятельности мы узнали много нового: основы языка программирования Python, основы Arduino IDE, мы узнали что такое компьютерное зрение, что такое нейронные сети, какие бывают нейронные сети и в чем их различия. Смогли реализовать свою собственную нейросеть и написать программу детектирования знаков при помощи компьютерного зрения. Это был совершенно новый опыт для нас, пришлось изучать все с нуля. Было много ошибок, но на ошибках учатся. Также мы смогли применить свои знания физики во время разработки электрической схемы.

В целом результат получился нормальный, но мы и дальше собираемся совершенствовать наш автомобиль.

Список литературы:

1) Stack Overflow - https://stackoverflow.com/

2) Документация OpenCV - https://docs.opencv.org/master/index.html

3) Документация Tensorflow - https://www.tensorflow.org/api_docs

4) Видеокурс по компьютерному зрению - https://avt.global/cv

5) Видеокурс по нейронным сетям - https://avt.global/neuralnets

6) Датасет - https://sim.newgen.education/

7) Документация Arduino - http://arduino.ru/Reference

8) Документация Keras - https://ru-keras.com/home/

9) Нейронная сеть - https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C

10) Нейросети для анализа изображений - https://www.youtube.com/watch?v=52U4BG0ENiM&list=PLtPJ9lKvJ4oi5ATzKmmp6FznCHmnhVoey