САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет прикладной математики-процессов управления
Кафедра компьютерных технологий и систем

Отчет по дисциплине «Цифровое управление подвижными объектами»

Выполнили: студенты группы18.М10-ПУ
Решетова Е. В., Сальникова М. В.
Проверил: Севостьянов Р. А.

Санкт-Петербург
2020

В рамках данной дисциплины перед нашей командой стояла следующая задача: необходимо разработать цифровое управление четырехколесным роботом при помощи микроконтроллера Arduino.

Этапы задачи:

1. Разработка алгоритма управления роботом при помощи микроконтроллера Arduino.
2. Реализация программы для микроконтроллера (скетч) в среде разработки Arduino IDE.
3. Разработка программы с графическим интерфейсом для управления работой микроконтроллера.
4. Разработка серверного приложения управления роботом.
5. Тестирование и исправление ошибок.
6. Написание отчета о проделанной работе.

Пункты 1, 2 данной работы были выполнены Сальниковой М. В., пункты 3 и 4 – Решетовой Е. В. 5 и 6 пункты выполнены совместно.

Первостепенной задачей была разработка алгоритма управления роботом. Управление происходило непосредственно при помощи микроконтроллера Arduino, к которому был присоединен двигатель с четырьмя колесами. Необходимо было реализовать возможность движения вперед, назад с разными скоростями, а также поворот.

Алгоритм для Arduino был написан в программе Arduino IDE. В нем были учтены все описанные ниже ограничения. Алгоритм заключался в переводе двух заданных чисел в пределах от -100 до 100 в числа, необходимые для корректного движения робота.

Мы решили, что с точки зрения простого пользователя удобнее всего задавать данные параметры при помощи показателей от -100 до 100 для поворота (-100 означает предельный поворот налево, 100 – направо, 0 для езды

прямо) и от -100 до 100 для значения скорости (-100 означает предел вращения колеса назад, 100 – вперед, 0 для остановки). Однако, необходимо совершить преобразования этих чисел, так как построенная схема и ограничения двигателя были таковы, что для управления движением было необходимо по 2 числа для каждой пары колес (правой и левой) – скорость и угол поворота, а также показатель движения назад. Кроме того, скорость и угол поворота в сумме не должны превышать показатель в 255 (5В).

Исходные данные	**Значение**
Скорость	[-100;100]
Поворот	[-100;100]

↓

Преобразованные данные	**Значение**	
Скорость (л.)	[0; 255]	Не более 255 в сумме
Поворот (л.)	[0; 255]	
Скорость (л.)	[0; 255]	Не более 255 в сумме
Поворот (пр.)	[0; 255]	
Движение вперед (л.)	0, 1	
Движение вперед (пр.)	0, 1	

Далее была реализована программа управления двигателем с графическим интерфейсом в среде программирования Qt на языке C++. Так как ограничения двигателя таковы, что сумма показателей скорости и поворота не должна превышать 255, сумма модулей изначальных данных, заданных пользователем, также не должна превышать предела в 100. Поэтому было принято решение реализовать графический интерфейс в виде ромба с координатами, где пользователь ставит точку на пересечении координат необходимых ему показателей скорости и поворота.

Следующим этапом стала разработка серверного приложения для обеспечения беспроводного управления. Сервер был добавлен непосредственно в программу управления двигателем с графическим интерфейсом, описанную в 3 этапе.

Для реализации сервера Qt предоставляет удобный класс *QTcpServer*, который предназначен для управления входящими TCP-соединениями. В предлагаемой программе объект класса *QTcpServer* будет слушать один из портов стека протоколов TCP/IP, по которому должен осуществляться нужный сервис, от всех хостов в сети. В нашем случае передается номер порта, равный 33333.

Для установки сервера нам необходимо вызвать в конструкторе метод *listen()*. При успешном вызове появляется сообщение об успешном запуске сервера. При возникновении ошибочных ситуаций, например, невозможности захвата порта, этот метод возвратит значение *false*, на которое мы отреагируем показом сообщения об ошибке.

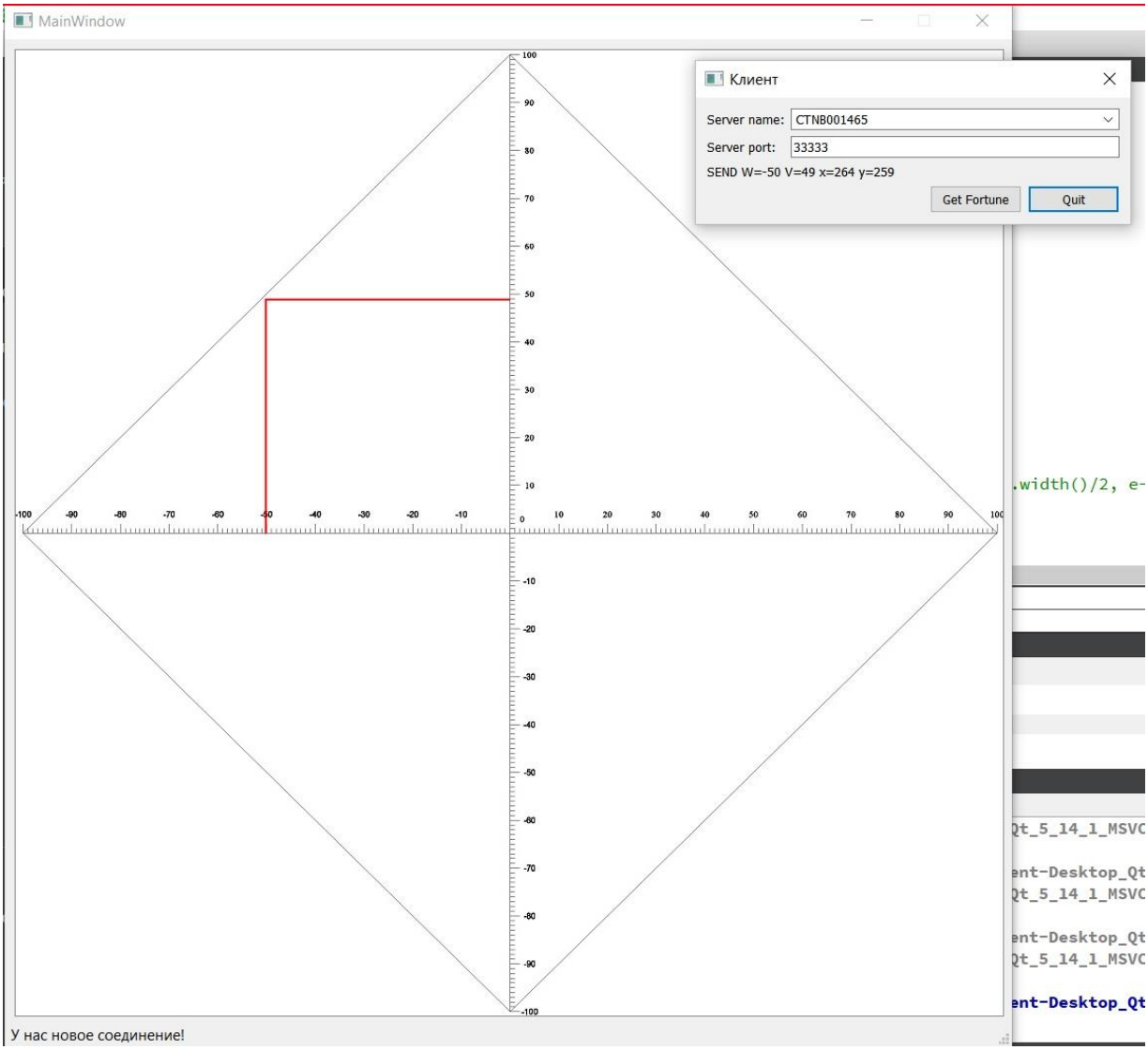
Далее мы производим соединение с сигналом *newConnection()*, который высылается при каждом присоединении нового клиента. Для подтверждения соединения с клиентом необходимо вызвать метод *nextPendingConnection()*, который возвратит сокет, посредством которого можно осуществлять дальнейшую связь с клиентом

Из метода *slotNewConnection()* выводится сообщение о присоединении нового клиента и выполняется передача координат, соответствующих показателям скорости и поворота.

Для проверки корректности передачи данных дополнительно был разработан клиент. Для реализации клиента необходимо создать объект класса *QTcpSocket*, а затем вызвать метод *connectToHost()*, передав в него первым параметром имя компьютера (или его IP-адрес), а вторым — номер порта сервера. Объект класса *QTcpSocket* сам попытается произвести установку связи с сервером и, в случае успеха, вышлет сигнал *connected()*.

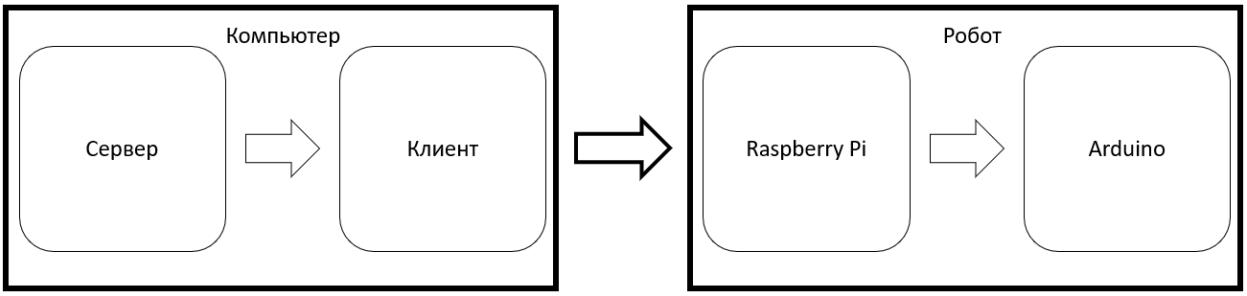
Слот *slotReadytoRead()* вызывается при поступлении данных от сервера, которые выводятся на визуальную форму.

Итоговая программа выглядит следующим образом:



В окне с графическим интерфейсом мы видим поставленную точку на пересечении координат (-50;49), что соответствует тому, что мощность двигателя распределена наполовину между скоростью (движением вперед) и поворотом направо. Эти данные передаются клиенту, после чего идут на обработку роботу.

Конечная схема передачи данных в таком случае может выглядеть следующим образом:



Таким образом, нашей команда было разработано цифровое управление четырехколесным роботом при помощи микроконтроллера Arduino, а также предложен способ дальнейшей работы данной системы.

Код программ расположен по ссылке в открытом доступе: https://github.com/ElenaReshetova/MachineControl