ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**МОРСКОЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**имени адмирала Г. И. Невельского**

Кафедра безопасности информации и телекоммуникационных систем

**Семестровый отчет**

Команды *студентов* 1-го курса

по выполнению проекта в рамках дисциплины НИРС ГСЭ

C*писок участников команды:*

1. Патенкова Анастастия (капитан)
2. Кондратьев Александр
3. Григорец Владислав
4. Фурман Кирилл
5. Гой Михаил
6. Буренок Дмитрий
7. Филиппов Дмитрий

*Руководитель*

*\_\_\_\_\_\_\_\_\_/Гончаров С.М./*

*«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2017 г.*

Владивосток

2017

СОДЕРЖАНИЕ

[Введение 4](#_Toc485770872)

[Компоненты: Raspberry Pi 3 Model B 5](#_Toc485770873)

[Основная информация 5](#_Toc485770874)

[Технические характеристки 6](#_Toc485770875)

[Компоненты: плата Arduino Uno R3 7](#_Toc485770876)

[Общие сведения 7](#_Toc485770877)

[Питание 8](#_Toc485770878)

[Выводы питания 8](#_Toc485770879)

[Память 8](#_Toc485770880)

[Входы и выходы 8](#_Toc485770881)

[Связь 9](#_Toc485770882)

[Токовая защита разъема USB 10](#_Toc485770883)

[Физические характеристики 10](#_Toc485770884)

[Программная составляющая (Arduino Uno) 10](#_Toc485770885)

[Программирование 10](#_Toc485770886)

[Автоматическая (программная) перезагрузка 10](#_Toc485770887)

[Компоненты: модуль, преобразователь понижающий DC-DC LM2596S 12](#_Toc485770888)

[Характеристики: 12](#_Toc485770889)

[Компоненты: драйвер шагового двигателя L298N 14](#_Toc485770890)

[Компоненты: сервопривод Tower Pro 9g SG90 15](#_Toc485770891)

[Основные технические характеристики 15](#_Toc485770892)

[Как подключать 15](#_Toc485770893)

[Схема подключения сервопривода 15](#_Toc485770894)

[Программирование сервопривода Tower Pro 15](#_Toc485770895)

[Функция управления скоростью сервопривода(servoslow) 15](#_Toc485770896)

[Сложности при работе с сервоприводами 16](#_Toc485770897)

[Работа над проектом 17](#_Toc485770898)

[Сборка робота 17](#_Toc485770899)

[Прошивка плат (Arduino и Raspberry) 18](#_Toc485770900)

[Запуск и отладка 29](#_Toc485770901)

Введение

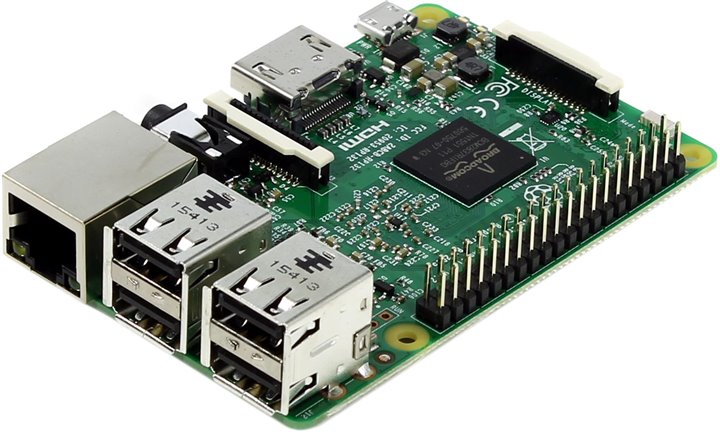
В начале дисциплины НИРС ГСЭ перед нами встал вопрос выбора темы нашего будущего проекта. После рассмотрения различных вариантов, мы решили, что сделаем робота на Raspberry Pi 3 Model B, который в свою очередь должен будет распознавать линии и впоследствии двигаться по ним.

Во многом в выполнении нашего проекта нам помогли работники Центра Развития Робототехники Гламаздин Юрий и Бажин Александр, которые могли проконсультировать нас по всем нашим вопросам по сборке робота.

Перед тем как начать покупку комплектующих и сборку из них робота, нам предстояло установить библиотеку OpenCV, с помощью которой и выполнялся весь проект. С установкой библиотеки возникало достаточно проблем, но все они решались по мере возникновения, по итоге спустя две недели после начала установки библиотеки на компьютеры всех участников команды мы имели 8 компьютеров с установленной OpenCV.

Примерный список покупок к концу проекта выглядел так:

1. Микрокомпьютер Raspberry Pi 3 Model B.
2. Плата Arduino UNO R3.
3. Модуль, преобразователь понижающий DC-DC LM2596S.
4. Драйвер шагового двигателя L298N.
5. Моторы и колёса.
6. Мини-сервопривод TowerPro SG-90.
7. Arduino соединительные провода для breadboard.
8. Камера.
9. Аккумуляторы и отсеки под них.
10. Выключатель.

Компоненты: Raspberry Pi 3 Model B

**Рис.1** *Микрокомпьютер Raspberry Pi 3 Model B*

***Основная информация***

Компактный ПК на базе единственной платы с новым, более мощным, четырехъядерным 64-битным процессором ARMv8 1,2ГГц, а также встроенными модулями Wi-Fi 802.11n и Bluetooth 4.1 с поддержкой BLE. Одна из главных возможностей - режим тонкого клиента с подключением к серверу по локальной сети с использованием протокола RDP. К другим особенностям можно отнести наличие выводов GPIO и поддержку HDMI CEC. Модель обладает оптимальной производительностью и справится со всеми поставленными задачами. Набор интерфейсов позволяет полноценно пользоваться собранной техникой и подключать к ней различные внешние устройства.

***Технические характеристки***

|  |  |
| --- | --- |
| Процессор компьютера | Broadcom BCM2837 ARMv8 Cortex-A53 (1.2 ГГц) |
| **Видео:** | |
| Тип видео | Встроенное |
| **Конфигурация:** | |
| Оперативная память | 1 ГБ LPDDR2 900 МГц |
| **Коммуникации:** | |
| Стандарты Wi-Fi | IEEE 802.11n (BCM43438 2.4 ГГц) |
| Интегрированный Bluetooth | Есть, Bluetooth 4.1 |
| Сеть | 10/100 Мбит/сек |
| **Интерфейс, разъемы и выходы:** | |
| Слоты для карт памяти | microSDHC, microSD |
| Разъемы компьютера | 4 x USB 2.0, HDMI,  RJ-45, 1 x USB microB, AV-out,  GPIO, разъем для камеры |
| **Охлаждение:** | |
| Охлаждение | Безвентиляторная система охлаждения |
| **Питание:** | |
| Потребление энергии | 0.5 - 1 Вт |
| **Сетевые характеристики:** | |
| Протоколы | RDP |

Компоненты: плата Arduino Uno R3

***Общие сведения***

Arduino Uno контроллер построен на ATmega328. Платформа имеет 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Для работы необходимо подключить платформу к компьютеру посредством кабеля USB, либо подать питание при помощи адаптера AC/DC или батареи.

В отличие от всех предыдущих плат, использовавших FTDI USB микроконтроллер для связи по USB, новый Ардуино Uno использует микроконтроллер ATmega8U2.

"Uno" переводится как «один» с итальянского, и разработчики тем самым намекают на грядущий выход Arduino 1.0. Новая плата стала флагманом линейки плат Ардуино.

|  |  |
| --- | --- |
| Микроконтроллер | ATmega328 |
| Рабочее напряжение | 5 В |
| Входное напряжение (рекомендуемое) | 7-12 В |
| Входное напряжение (предельное) | 6-20 В |
| Цифровые Входы/Выходы | 14 (6 из которых могут использоваться как  выходы [ШИМ)](http://arduino.ru/Tutorial/PWM) |
| Аналоговые входы | 6 |
| Постоянный ток через вход/выход | 40 мА |
| Постоянный ток для вывода 3.3 В | 50 мА |
| Флешпамять | 32 Кб (ATmega328) из которых 0.5 Кб используются для загрузчика |
| ОЗУ | 2 Кб (ATmega328) |
| EEPROM | 1 Кб (ATmega328) |
| Тактовая частота | 16 МГц |

**Табл.1** *Характеристики платы*

***Питание***

Arduino Uno может получать питание через подключение USB или от внешнего источника питания. Источник питания выбирается автоматически.

Внешнее питание (не USB) может подаваться через преобразователь напряжения AC/DC (блок питания) или аккумуляторной батареей. Преобразователь напряжения подключается посредством разъема 2.1 мм с центральным положительным полюсом. Провода от батареи подключаются к выводам Gnd и Vin разъема питания.

Платформа может работать при внешнем питании от 6В до 20В. При напряжении питания ниже 7В, вывод 5V может выдавать менее 5В, при этом платформа может работать нестабильно. При использовании напряжения выше 12В регулятор напряжения может перегреться и повредить плату. Рекомендуемый диапазон от 7В до 12В.

***Выводы питания***

* VIN. Вход используется для подачи питания от внешнего источника (в отсутствие 5В от разъема USB или другого регулируемого источника питания). Подача напряжения питания происходит через данный вывод.
* 5V. Регулируемый источник напряжения, используемый для питания микроконтроллера и компонентов на плате. Питание может подаваться от вывода VIN через регулятор напряжения, или от разъема USB, или другого регулируемого источника напряжения 5В.
* 3V3. Напряжение на выводе 3,3В генерируемое встроенным регулятором на плате. Максимальное потребление тока 50 мА.
* GND. Выводы заземления.

***Память***

Микроконтроллер ATmega328 располагает 32 Кб флэш памяти, из которых 0.5 кБ используется для хранения загрузчика, а также 2 Кб ОЗУ (SRAM) и 1 Кб EEPROM. (которая читается и записывается с помощью [библиотеки EEPROM)](http://arduino.ru/Reference/Library/EERPOM).

***Входы и выходы***

Каждый из 14 цифровых выводов Uno может настроен как вход или выход, используя функции [pinMode(),](http://arduino.ru/Reference/PinMode) [digitalWrite(),](http://arduino.ru/Reference/DigitalWrite) и [digitalRead().](http://arduino.ru/Reference/DigitalRead) Выводы работают при напряжении 5 В. Каждый вывод имеет нагрузочный резистор (по умолчанию отключен) 20-50 кОм и может пропускать до 40 мА. Некоторые выводы имеют особые функции:

* Последовательная шина: 0 (RX) и 1 (TX). Выводы используются для получения (RX) и передачи (TX) данных TTL. Данные выводы подключены к соответствующим выводам микросхемы последовательной шины ATmega8U2 USB-to-TTL.
* Внешнее прерывание: 2 и 3. Данные выводы могут быть сконфигурированы на вызов прерывания либо на младшем значении, либо на переднем или заднем фронте, или при изменении значения.

Подробная информация находится в описании функции [attachInterrupt()](http://arduino.ru/Reference/AttachInterrupt).

* ШИМ: 3, 5, 6, 9, 10, и 11. Любой из выводов обеспечивает [ШИМ](http://arduino.ru/Tutorial/PWM) с разрешением 8 бит при помощи функции [analogWrite()](http://arduino.ru/Reference/AnalogWrite).
* SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Посредством данных выводов осуществляется связь SPI, для чего используется [библиотека SPI](http://arduino.ru/Reference/Library/SPI).
* LED: 13. Встроенный светодиод, подключенный к цифровому выводу 13. Если значение на выводе имеет высокий потенциал, то светодиод горит.

На платформе Uno установлены 6 аналоговых входов (обозначенных как A0 - A5), каждый разрешением 10 бит (т.е. может принимать 1024 различных значения). Стандартно выводы имеют диапазон измерения до 5 В относительно земли, тем не менее имеется возможность изменить верхний предел посредством вывода AREF и функции [analogReference().](http://arduino.ru/Reference/AnalogReference) Некоторые выводы имеют дополнительные функции:

* I2C: 4 (SDA) и 5 (SCL). Посредством выводов осуществляется связь I2C (TWI), для создания которой используется [библиотека Wire](http://arduino.cc/en/Reference/Wire).

Дополнительная пара выводов платформы:

* AREF. Опорное напряжение для аналоговых входов. Используется с функцией [analogReference()](http://arduino.ru/Reference/AnalogReference).
* Reset. Низкий уровень сигнала на выводе перезагружает микроконтроллер. Обычно применяется для подключения кнопки перезагрузки на плате расширения, закрывающей доступ к кнопке на самой плате Arduino.

***Связь***

На платформе Arduino Uno установлено несколько устройств для осуществления связи с компьютером, другими устройствами Arduino или микроконтроллерами. ATmega328 поддерживают последовательный интерфейс UART TTL (5В), осуществляемый выводами 0 (RX) и 1 (TX). Установленная на плате микросхема ATmega8U2 направляет данный интерфейс через USB, программы на стороне компьютера "общаются" с платой через виртуальный COM порт. Прошивка ATmega8U2 использует стандартные драйвера USB COM, никаких сторонних драйверов не требуется, но на Windows для подключения потребуется файл ArduinoUNO.inf. Мониторинг последовательной шины (Serial Monitor) программы Arduino позволяет посылать и получать текстовые данные при подключении к платформе. Светодиоды RX и TX на платформе будут мигать при передаче данных через микросхему FTDI или USB подключение (но не при использовании последовательной передачи через выводы 0 и 1).

Библиотекой SoftwareSerial возможно создать последовательную передачу данных через любой из цифровых выводов Uno.

ATmega328 поддерживает интерфейсы I2C (TWI) и SPI. В Arduino включена библиотека Wire для удобства использования шины I2C.

***Токовая защита разъема USB***

В Arduino Uno встроен самовосстанавливающийся предохранитель (автомат), защищающий порт USB компьютера от токов короткого замыкания и сверхтоков. Хотя практически все компьютеры имеют подобную защиту, тем не менее, данный предохранитель обеспечивает дополнительный барьер. Предохранитель срабатывает при прохождении тока более 500 мА через USB порт и размыкает цепь до тех пока нормальные значения токов не будут восстановлены.

***Физические характеристики***

Длина и ширина печатной платы Uno составляют 6.9 и 5.3 см соответственно. Разъем USB и силовой разъем выходят за границы данных размеров. Четыре отверстия в плате позволяют закрепить ее на поверхности. Расстояние между цифровыми выводами 7 и 8 равняется 0,4 см, хотя между другими выводами оно составляет 0,25 см.

***Программная составляющая (Arduino Uno)***

*Программирование*

Платформа программируется посредством ПО Arduino. Из меню Tools > Board выбирается «Arduino Uno» (согласно установленному микроконтроллеру.

Микроконтроллер ATmega328 поставляется с записанным загрузчиком, облегчающим запись новых программ без использования внешних программаторов. Связь осуществляется оригинальным протоколом STK500.

Имеется возможность не использовать загрузчик и запрограммировать микроконтроллер через выводы ICSP (внутрисхемное программирование).

*Автоматическая (программная) перезагрузка*

Uno разработана таким образом, чтобы перед записью нового кода перезагрузка осуществлялась самой программой Arduino на компьютере, а не нажатием кнопки на платформе. Одна из линий DTR микросхемы ATmega8U2, управляющих потоком данных (DTR), подключена к выводу перезагрузки микроконтроллеру ATmega328 через 100 нФ конденсатор. Активация данной линии, т.е. подача сигнала низкого уровня, перезагружает микроконтроллер. Программа Arduino, используя данную функцию, загружает код одним нажатием кнопки Upload в самой среде программирования. Подача сигнала низкого уровня по линии DTR скоординирована с началом записи кода, что сокращает таймаут загрузчика.

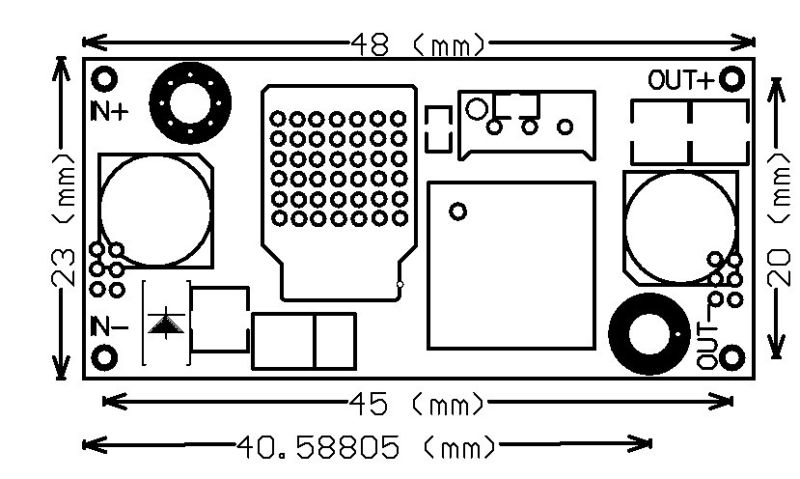
Функция имеет еще одно применение. Перезагрузка Uno происходит каждый раз при подключении к программе Arduino на компьютере с ОС Mac X или Linux (через USB). Следующие полсекунды после перезагрузки работает загрузчик. Во время программирования происходит задержка нескольких первых байтов кода во избежание получения платформой некорректных данных (всех, кроме кода новой программы). Если производится разовая отладка скетча, записанного в платформу, или ввод каких-либо других данных при первом запуске, необходимо убедиться, что программа на компьютере ожидает в течение секунды перед передачей данных.

На Uno имеется возможность отключить линию автоматической перезагрузки разрывом соответствующей линии. Контакты микросхем с обоих концов линии могут быть соединены с целью восстановления. Линия маркирована «RESET-EN». Отключить автоматическую перезагрузку также возможно подключив резистор 110 Ом между источником 5 В и данной линией.

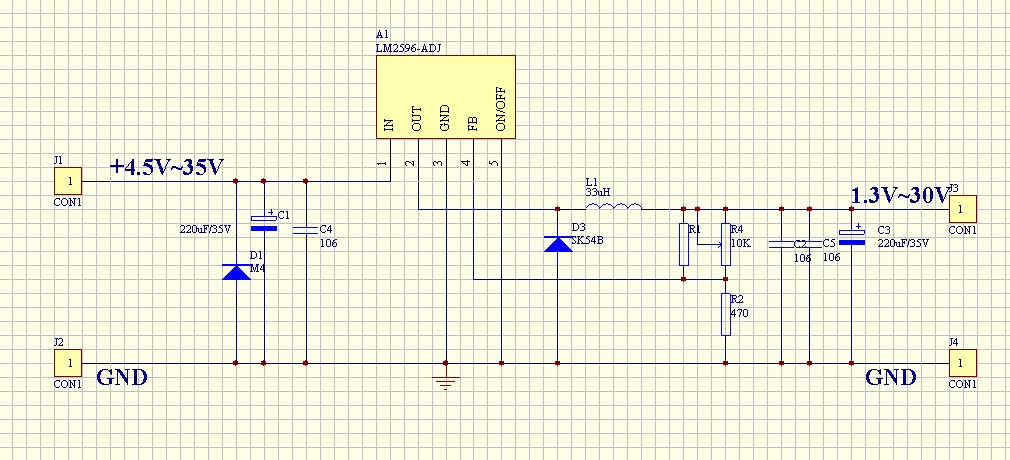
Компоненты: модуль, преобразователь понижающий DC-DC LM2596S

Импульсный понижающий регулируемый стабилизатор постоянного напряжения. Имеет высокий КПД. Меньше нагревается если сравнивать с модулями на линейных стабилизаторах. Источник питания может применяться в широком спектре устройств. К безусловным достоинствам относится работа в ощутимом диапазоне входного напряжения. Вместе с большим КПД это дает хорошие результаты при последовательном включении DC–DC LM2596 с химическими источниками тока, солнечными панелями или ветряными генераторами. Дополнив преобразователь DC–DC LM2596 трансформатором, выпрямителем и фильтром получим блок питания. На входе стабилизатора напряжение должно быть большее выходного минимум на 1,5 В. При потреблении мощности от DC–DC LM2596 более десяти Вт следует применять средства охлаждения.

*Характеристики:*

1. Входное напряжение 4,5–28 B
2. Выходное напряжение 1,3–25 B
3. Наибольший уровень шумов 50 мВ
4. Влияние изменения входного напряжения на уровень выхода 0,5 %
5. Поддержание установленного напряжения с точностью 2,5 %
6. Выходной ток: номинальный до 1 А, от 1 до 2 А заметно возрастает нагрев, предельный 3 А
7. Частота преобразования 150 КГц
8. КПД до 92%
9. Пределы температуры окружающего воздуха во время работы40-85 °С

**Рис.2** *Схема преобразователя*

Нестабилизированное напряжение подается на контакты модуля +IN, –IN. На входе установлены: ограничитель всплесков напряжения D1, конденсаторы C1 и С4 сглаживающие пульсации в напряжении, поступающем на модуль. Также эти компоненты снижают шумы, поступающие в общую сеть питания. Микросхема [LM2596S-ADJ](https://arduino-kit.ru/userfiles/image/LM2596.pdf) содержит мощный транзистор управляющий выходным напряжением в импульсном режиме.

**Рис.3** *Электрическая схема DC–DC LM2596*

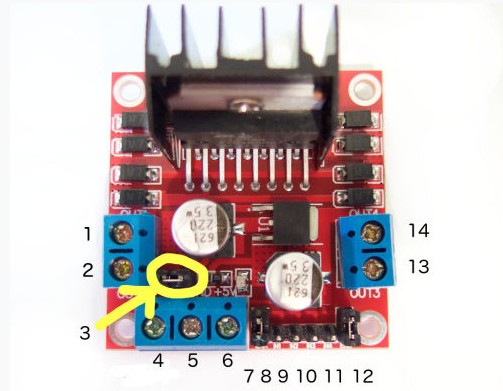
Выход микросхемы OUT вывод 2 подключается внутренним транзистором на короткое время к входу микросхемы – импульс. За это время происходит накопление энергии в индуктивности L1. Затем транзистор МС полностью отключает выход МС и энергия, запасенная в индуктивности, передается потребителю. Когда транзистор МС закрыт, то ток питания нагрузки проходит по цепи, в которую входит диод D3.

Так как частота коммутации высокая, то требуется небольшая индуктивность для такой схемы. Резисторы позволяют организовать петлю обратной связи подключенную к входу микросхемы FB. Для точной установки выходного напряжения преобразователь DC–DC LM2596 имеет многооборотный переменный резистор. Благодаря обратной связи, контролирующей выходное напряжение, происходит стабилизация. Конденсаторы C2, C3, C5 на выходе схемы снижают шумы в напряжении источника до незначительной величины. Выходное напряжение снимается с контактов платы +OUT, -OUT.

Высокий КПД и сниженный нагрев объясняется следующими причинами. Питание коммутируется мощным транзистором микросхемы, работающим в двух режимах: полностью открыт или закрыт. Сопротивление открытого транзистора очень низкое, при протекании через него тока нагрев мал. Через закрытый транзистор ток не течет – нагрева нет. D4 – диод Шотки, на нем малое падение напряжения и следовательно D4 почти не нагревается.

Стабилизация выходного напряжения происходит путем изменения скважности импульсов, управляющих транзистором. Для этого в микросхеме имеется ШИМ генератор, изменяющий скважность импульсов под действием напряжения, поступающего на вход 4 микросхемы. Частота генератора постоянна.

Компоненты: драйвер шагового двигателя L298N



**Рис.4**

1. Для двигателя постоянного тока 1 “+” или для шагового двигателя A+
2. Для двигателя постоянного тока 1 “-” или для шагового двигателя A-
3. Коннектор на 12В. Снимите его, если используете напряжение питания больше 12В
4. Питания вашего двигателя обеспечивается с этого выхода. Максимальное напряжение питания постоянным током 35 вольт. Если напряжение больше 12 вольт, разомкните контакты на 3 коннекторе.
5. GND - земля
6. Питание 5В, если коннектор на 12В замкнут. Идеально для питания Arduino и т.п.
7. Коннектор для двигателя постоянного тока 1. Можно подключить к ШИМ-выходу для управления скоростью двигателя постоянного тока
8. IN1
9. IN3
10. IN4
11. Коннектор для двигателя постоянного тока 2. В случае использования шагового двигателя, подключать сюда ничего не надо. Можно подключить к ШИМ-выходу для управления скоростью двигателя постоянного тока
12. Двигатель постоянного тока 2 “+” или шаговый двигатель B+
13. Двигатель постоянного тока 2 “-” или шаговый двигатель B-
14. IN2

Компоненты: сервопривод Tower Pro 9g SG90

***Основные технические характеристики***

* Материал редуктора: Нейлон
* Напряжение питания: 3 - 5В
* Усилие на валу: 1.2кг/см (4.8В); 1,6кг/см (6,0В)
* Мертвая ширина импульса: 2мс
* Размер: Микро (22мм х 11.5мм х 27мм)
* Масса: 9г
* Угол поворота: 160°

• Температура использования: от -30 до +60 градусов по Цельсию

***Как подключать***

Сервопривод Tower Pro 9g SG90 используется в основном для управления небольшими легкими механизмами, угол поворота которых ограничен диапазоном от 0 до 180 градусов.

***Схема подключения сервопривода***

* Коричневый провод – Земля (Ground, подключается к пину GND на плате Arduino)
* Красный провод – Питание +5 V (подключается к пину 5V на плате Arduino)
* Желтый провод – Сигнал управления (подключается к цифровому пину Arduino)

***Программирование сервопривода Tower Pro***

Необходимо в первую очередь подключить библиотеку Servo, которая облегчает работу с различными сервоприводами и значительно упрощает программный код. Затем, объявить сервопривод. Далее внутри исполняемого блока программы управлять положением сервопривода.

***Функция управления скоростью сервопривода(servoslow)***

Функция управления скоростью сервопривода использует следующие параметры:

1. Имя сервопривода
2. Конечная позиция сервопривода (в градусах)
3. Задержка между каждым поворотом в один градус (чем больше значение, тем медленнее будет поворачиваться сервопривод)
4. Начальная позиция сервопривода

***Сложности при работе с сервоприводами***

Недорогие сервоприводы не обеспечивают точных настроек начальной и конечной позиций. Часто в положении 0 и 180 градусов слышен характерный треск или жужжание. Это говорит о том, что механическая часть сервопривода уже находиться в предельной позиции, но датчик считает, что заданная позиция еще не достигнута. Проблему можно решить подбором скорректированных позиций. Например, часто вместо позиции 0 градусов нужно использовать позицию поворота сервопривода в 10-15 градусов, а конечную 165-170 градусов. Это позволит избежать лишних перегрузок и шума при достижении крайних точек.

Следует обращать внимание на напряжение, которое используется при работе сервопривода. Если значение напряжения превышает допустимые значения из технических характеристик сервопривода, механические части могут выйти из строя или зубчатые колеса сервопривода провернутся в положение превышающее максимальные значения и механизм не сможет продолжать движение в обратном направлении. Иными словами, завышенное напряжение скорее всего не приведет к сгоранию мотора сервопривода, но повредит механические части зубчатых механизмов.

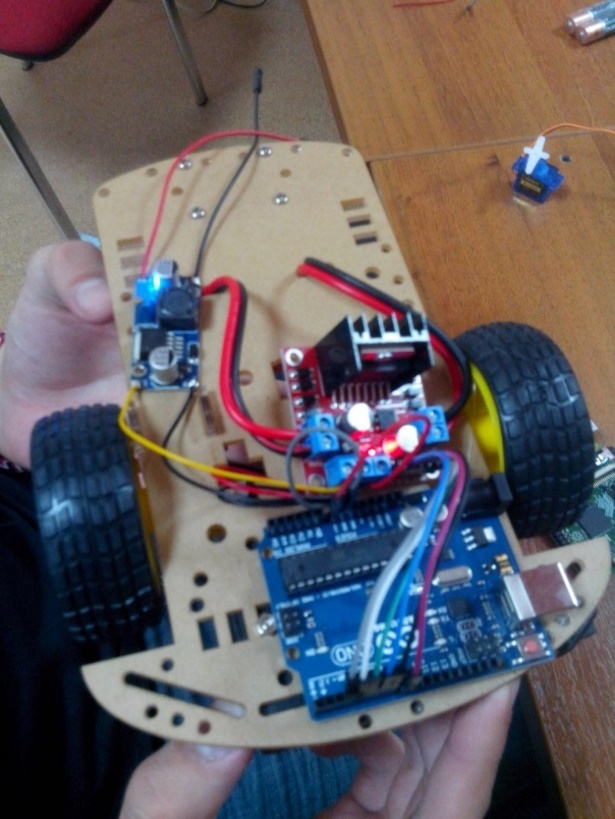
Работа над проектом

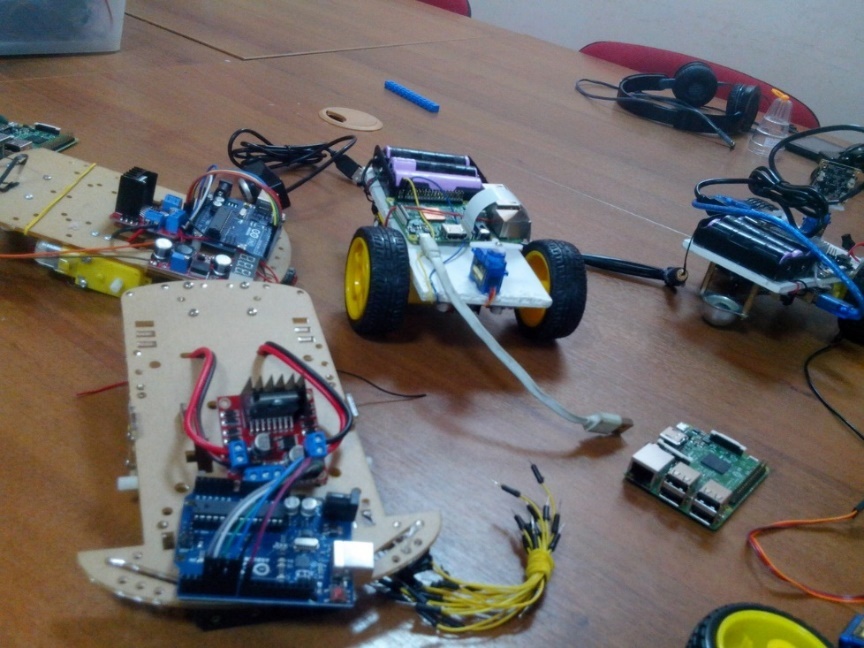
***Сборка робота***

*Закупка деталей из списка:*

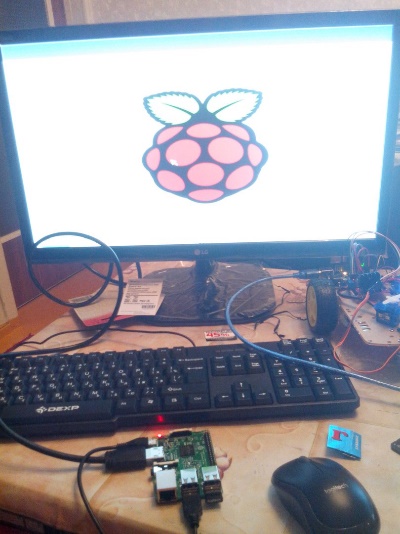
1. Микрокомпьютер Raspberry Pi 3 Model B.
2. Плата Arduino UNO R3.
3. Модуль, преобразователь понижающий DC-DC LM2596S.
4. Драйвер шагового двигателя L298N.
5. Моторы и колёса.
6. Мини-сервопривод TowerPro SG-90.
7. Arduino соединительные провода для breadboard.
8. Камера.
9. Аккумуляторы и отсеки под них.
10. Выключатель.

**Рис.5** *Закупленные компоненты*

*Сборка робота из закупленных компонентов в ЦРР (Центре Развития Робототехники) совместно с командами Центра:*

**Рис.6** *Робот на этапе сборки* **Рис.7** *Роботы (наш и экземпляры ЦРР)*

***Прошивка плат (Arduino и Raspberry)***

Для работы с микрокомпьютером Raspberry Pi 3 Model B было необходимо для начала установить на него операционную систему Raspbian (бесплатная операционная система для плат Raspberry на базе Debian), а после – скомпилировать и установить библиотеку OpenCV для языка СИ, а также библиотеку для последовательного порта LibSerial (она необходима, т.к. взаимодействие между исполняемым кодом на микрокомпьютере и кодом нижнего уровня осуществляется через serial port (последовательный порт).

**Рис.8** *Плата Raspberry на этапе прошивки* **Рис.9** *Загрузка компьютера Raspberry*

Программный код верхнего уровня (основной код, в котором используется алгоритм компьютерного зрения и библиотека OpenCV) компилируется и выполняется именно на микрокомпьютере Raspberry. Для компиляции кода необходимы файлы:

1. Source file (текстовый файл с программным кодом на языке СИ с уже подключенной библиотекой OpenCV)
2. Makefile (набор инструкций для программы make, которая помогает собирать программный проект)

Итак, содержание файла ресурсов ***test.cpp***:

#include <cv.h>// подключение заголовочных файлов библиотеки OpenCV

#include <highgui.h>

#include <imgproc.hpp>

#include <video.hpp>

#include <stdlib.h>

#include <stdio.h>

#include <SerialStream.h>

using namespace LibSerial ;

using namespace std;

using namespace cv;

char device[]= "/dev/ttyACM0";

SerialStream my\_serial\_stream;

IplImage\* rgb = 0;

void MakeCom()

{

my\_serial\_stream.Open(device) ;

my\_serial\_stream.SetBaudRate( SerialStreamBuf::BAUD\_115200 ) ;

cout << "connect " << endl;

}

void SendToSerialMove(int m1, int m2, int t) {

my\_serial\_stream << "|MOVE," << m1 << "," << m2 << "," << t << "|";

usleep(2000);

}

void SendToSerialServa(int s1) {

my\_serial\_stream << "|SERVA" << s1 << "|";

usleep(2000);

}

int main(int argc, char\* argv[])

{

MakeCom();

CvCapture\* capture = cvCreateCameraCapture(CV\_CAP\_ANY);

assert( capture );

double width = cvGetCaptureProperty(capture, CV\_CAP\_PROP\_FRAME\_WIDTH);

double height = cvGetCaptureProperty(capture, CV\_CAP\_PROP\_FRAME\_HEIGHT);

IplImage\* frame=0;

IplImage \*gray=0;

int k;

while(true){

frame = cvQueryFrame( capture );

gray = cvCreateImage(cvGetSize(frame),IPL\_DEPTH\_8U,1);

cvCvtColor(frame,gray,CV\_RGB2GRAY);

rgb = cvCreateImage( cvGetSize(gray), IPL\_DEPTH\_8U, 0 );

cvCopy(gray, rgb);

cvInRangeS(gray, cvScalar(0, 0, 0), cvScalar(65, 65, 65), rgb);

int y=rgb->height-5;

int a[rgb->width] = {0};

int pos=-1;

for( int x=0; x<=rgb->width; x++ )

{

k = CV\_IMAGE\_ELEM(rgb, uchar, y, x);

if(k==255){pos+=1; a[pos]=x;}

}

if(pos!=-1)

{

int MIDDLE=(rgb->width)/2;

int LEFT=MIDDLE-a[0];

int RIGHT=a[pos]-MIDDLE;

if(LEFT>80 && RIGHT>80){SendToSerialMove(-150,-140,25);}

if(LEFT>40 && LEFT<80){SendToSerialMove(-75,-140,25);}

if(RIGHT>40 && RIGHT<80){SendToSerialMove(-150,-75,25);}

if(LEFT<40 && RIGHT>0){SendToSerialMove(0,-140,25);}//right turn

if(LEFT>0 && RIGHT<40){SendToSerialMove(-150,0,25);}//left turn

}

char c = cvWaitKey(33);

if (c == 27) { // если нажата ESC - выходим

break;

}

//cvShowImage("rgb and", rgb); - показывает обрабатываемое изображение, нужна для отладки цветовой гаммы, используется при настройке кода при подключении к Raspberry монитора

}

cvReleaseCapture( &capture );

cvReleaseImage(& frame);

cvReleaseImage(& gray);

cvReleaseImage(& rgb);

//cvDestroyAllWindows();

return 0;

}

Содержание файла настроек компилятора (Makefile):

CC := g++

CFLAGS := -L/usr/local/lib -L/usr/lib -I/usr/local/include -I/usr/local/include/opencv2/videoio -I/usr/local/include/opencv -I/usr/local/include/opencv2/highgui -I/usr/local/include/opencv2/core -I/usr/local/include/opencv2

OBJECTS :=

LIBRARIES := -lserial -lopencv\_core -lopencv\_imgproc -lopencv\_highgui -lopencv\_imgcodecs -lopencv\_videoio

PKG := pkg-config –cflags –libs libserial

.PHONY: all clean

all: test

test:

$(CC) $(CFLAGS) -o test test.cpp $(LIBRARIES)

clean:

rm -f \*.o

*Процесс сборки проекта:*

Компилятор (g++) берет файлы с исходным кодом и получает из них объектные файлы. Затем линковщик берет объектные файлы и получает из них исполняемый файл. Сборка = компиляция + линковка.

В командной строке нужно набрать команды:

1. **cd** /*путь до папки, где расположены оба вышеуказанных файла/*
2. **make** (в результате успешного выполнения команды создаётся исполняемый файл программы, в противном случае – на консоль выводится информация об ошибке)
3. **./test** (шаблон: *./имя\_исполняемого\_файла*, команда для запуска программы)

После окончания прошивки плата крепится на робота, к ней подключается камера и плата Arduino, запитывается Raspberry через DC/DC-преобразователь от четырёх аккумуляторов 18600.

На плату Arduino посредством соединения её с ноутбуком загружается программный код нижнего уровня, состоящий из трёх компонентов:

1. OPENCV\_ARDUINO.ino – Arduino file
2. QList.cpp
3. QList.h

Содержание файла OPENCV\_ARDUINO.ino:

#define SPEED\_1 6 // 5

#define DIR\_1 7 // 3

#define DIR\_2 8 // 4

#define SPEED\_2 5 //6

#define DIR\_3 4 //8

#define DIR\_4 3 //7

#include <Servo.h>

Servo myServo;

#include "QList.h"

// Need to be for unknown reason

#include "QList.cpp"

QList<String> myList;

double start = millis();

int timer = 0;

bool m = false;

String inputString = "";

void Stop()

{

analogWrite(SPEED\_1, 0);

analogWrite(SPEED\_2, 0);

digitalWrite(DIR\_1, LOW);

digitalWrite(DIR\_2, LOW);

digitalWrite(DIR\_3, LOW);

digitalWrite(DIR\_4, LOW);

}

void Move(int speed\_motor1, int speed\_motor2)

{

//Функция движения. если скорость мотора отрицательная, то крутим в назад

// Для коллекторного мотора можно выбрать значение скорости от 0 до 255.

if (speed\_motor1 < 0)

{

digitalWrite(DIR\_1, LOW);

digitalWrite(DIR\_2, HIGH);

analogWrite(SPEED\_1, -speed\_motor1);

}

else

{

digitalWrite(DIR\_2, LOW);

digitalWrite(DIR\_1, HIGH);

analogWrite(SPEED\_1, speed\_motor1);

}

if (speed\_motor2 < 0)

{

digitalWrite(DIR\_3, HIGH);

digitalWrite(DIR\_4, LOW);

analogWrite(SPEED\_2, -speed\_motor2);

} else

{

digitalWrite(DIR\_4, HIGH);

digitalWrite(DIR\_3, LOW);

analogWrite(SPEED\_2, speed\_motor2);

}

}

void MoveT(int speed\_motor1, int speed\_motor2, double t)

{

// Движение по линии заданное количество миллисекунд

start = millis();

do

{

Move(speed\_motor1, speed\_motor2);

} while (start + t > millis());

Stop();

Serial.println("TIME LINE " + String(t) );

}

void serialEvent() {

bool read\_flag = false;

while (Serial.available()) {

// get the new byte:

char inChar = (char)Serial.read();

// add it to the inputString:

inputString += inChar;

// if the incoming character is a newline, set a flag

// so the main loop can do something about it:

if (inChar == '|') {

// помещаем пакет в очередь

String s = inputString;

myList.push\_back(s);

inputString = "";

//

/\*

// чистим остаток

while (Serial.available()) {

// get the new byte:

char inChar = (char)Serial.read();

}

\*/

}

//stringComplete = true;

}

if (read\_flag)

{

read\_flag = false;

}

}

void setup() {

// initialize serial:

Serial.begin(115200);

// reserve 200 bytes for the inputString:

inputString.reserve(1000);

pinMode(SPEED\_1, OUTPUT);

pinMode(DIR\_1, OUTPUT);

pinMode(SPEED\_2, OUTPUT);

pinMode(DIR\_2, OUTPUT);

pinMode(DIR\_3, OUTPUT);

pinMode(DIR\_3, OUTPUT);

myServo.attach(9);

myServo.write(90); // открыли захват

}

void loop() {

//delay(2000);

// MoveT(120, 100, 50);

// print the string when a newline arrives:

if (myList.size() > 0)

{

// if (stringComplete) {

// Serial.println(inputString);

// берем пакет из очереди

String strin = myList.at(0);

myList.pop\_front();

int pos = strin.indexOf("MOVE");

if ( pos >= 0)

{

// парсим пакет на 3 переменных

String S = strin.substring(pos + 5);

int pos2 = S.indexOf(",");

int spd1 = S.substring(0, pos2).toInt();

String S2 = S.substring(pos2 + 1);

int pos3 = S2.indexOf(",");

int spd2 = S2.substring(0, pos3).toInt();

String S3 = S2.substring(pos3 + 1);

timer = S3.toInt();

Serial.println("MOVE KOMANDA " + String(spd1) + " " + String(spd2) + " " + String(timer));

//Send("OK", ch\_id);

//if (tmr > 0)

// MoveT(spd1, spd2, tmr);

start = millis();

m = true;

Move(spd1, spd2);

// Образец команды

//|MOVE,50,50,500\n

}

pos = strin.indexOf("SERV");

if ( pos >= 0)

{

// парсим пакет на 3 переменных

String S = strin.substring(pos + 5);

int pos2 = S.indexOf("|");

int spd1 = S.substring(0, pos2).toInt();

//Serial.println("SRERV " + String(spd1));

int srv = 0;

if (spd1 < 0)

myServo.write(90 + spd1 \* -1);

if (spd1 >= 0)

myServo.write(90 - spd1);

}

// clear the string:

//stringComplete = false;

}

//Serial.println("STOP " + String(start) + " " + String(timer) + " " + String(millis()));

if (m == true)

if ((start + timer) < millis())

{

m = false;

//Serial.println("STOP " + String(start) + " " + String(timer) + " " + String(millis()));

Stop();

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание файла QList.cpp:

#include "QList.h"

// Constructor

template<typename T>

QList<T>::QList()

{

len = 0;

start = NULL;

end = NULL;

}

// Destructor

template<typename T>

QList<T>::~QList()

{

clear();

}

// Push at front

template<typename T>

void QList<T>::push\_front(const T i)

{

node \*tmp = new node;

tmp->item = i;

tmp->next = NULL;

tmp->prev = NULL;

if (start == NULL) // If list is empty

{

start = tmp;

end = tmp;

}

else // Insert at start

{

tmp->next = start;

start->prev = tmp;

start = tmp;

}

len++; // Increase size counter

}

// Push at back

template<typename T>

void QList<T>::push\_back(const T i)

{

node \*tmp = new node;

tmp->item = i;

tmp->next = NULL;

tmp->prev = NULL;

if (end == NULL) // If list is empty

{

start = tmp;

end = tmp;

}

else // Insert at the end

{

tmp->prev = end;

end->next = tmp;

end = tmp;

}

len++; // Increase size counter

}

// Pop from front

template<typename T>

void QList<T>::pop\_front()

{

if (start != NULL)

{

node \*tmp = start;

start = start->next;

if (start != NULL) // Re-link next item to NULL

start->prev = NULL;

else // List became empty so we need to clear end

end = NULL;

delete tmp;

len--; // Decrease counter

}

}

// Pop from back

template<typename T>

void QList<T>::pop\_back()

{

if (end != NULL)

{

node \*tmp = end;

end = end->prev;

if (end != NULL) //Re-link previous item to NULL

end->next = NULL;

else // List became empty so we need to clear start

start = NULL;

len--; // Decrease counter

}

}

// Get item from front

template<typename T>

T QList<T>::front()

{

if (start != NULL)

return start->item;

//TODO: Catch error when list is empty

}

//Get item from back

template<typename T>

T QList<T>::back()

{

if (end != NULL)

return end->item;

//TODO: Catch error when list is empty

}

// Get size

template<typename T>

int QList<T>::size()

{

return this->len;

}

// Clear list

template<typename T>

void QList<T>::clear()

{

node \*tmp = start;

while (start != NULL)

{

tmp = start;

start = start->next;

delete tmp;

len--; // Decrease counter

}

end = NULL;

}

template<typename T>

void QList<T>::clear(int index)

{

node \*tmp = start;

for (int i = 0;i <= index&&tmp != NULL;i++)

{

if (i == index)

{

if (tmp->prev != NULL)

tmp->prev->next = tmp->next;

else

start = tmp->next;

if (tmp->next != NULL)

tmp->next->prev = tmp->prev;

else

end = tmp->prev;

len--; // Decrease counter

break;

}

}

}

// Get at index

template<typename T>

T QList<T>::get(int index)

{

node \*tmp = start;

for (int i = 0;i <= index&&tmp != NULL;i++)

{

if (i == index)

return tmp->item;

else

tmp = tmp->next;

}

//TODO: Catch error when index is out of range

}

template<typename T>

T QList<T>::at(int index)

{

node \*tmp = start;

for (int i = 0;i <= index&&tmp != NULL;i++)

{

if (i == index)

return tmp->item;

else

tmp = tmp->next;

}

//TODO: Catch error when index is out of range

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание файла QList.h:

#ifndef LIB\_SCP\_QLIST

#define LIB\_SCP\_QLIST

#ifndef NULL

#define NULL 0

#endif

template<typename T>

class QList

{

private:

typedef struct node

{

T item;

node \*next, \*prev;

}node;

int len; //!< Size of list >

node \*start, \*end; //!< Pointers to start and end >

public:

QList(); //!< Class constructor >

~QList(); //!< Class destructor >

void push\_back(const T i); //!< Push item at the back of list >

void push\_front(const T i);//!< Push item at the front of the list >

void pop\_back(); //!< Pops item from back >

void pop\_front(); //!< Pops item from front >

T front(); //!< get item from front >

T back(); //!< get item from back >

int size(); //!< Returns size of list >

void clear(); //!< Clears list >

void clear(int index); //!< Clears list >

T get(int index); //!< Get item at given index >

T at(int index); //!< Get item at given index >

};

#endif

Код верхнего уровня сообщается с кодом нижнего через отправку пакетов данных формата:

1. |MOVE,500,500,1000\n
2. |SERV 90\n

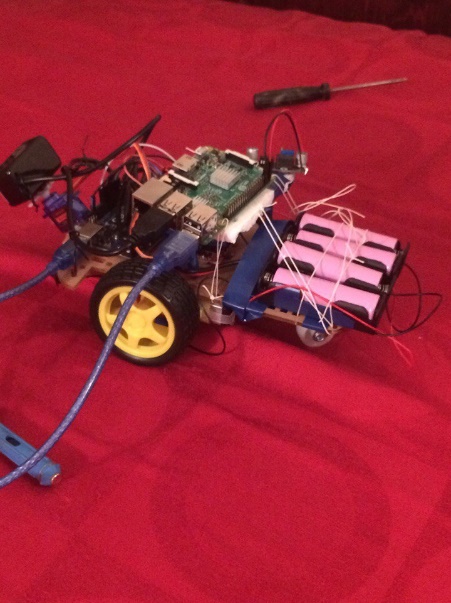
Первая команда – движение робота вперёд/назад (первые два аргумента функции – напряжения на левый и правый моторы соответственно (со знаком + или – в зависимости от направления), третье число – время в миллисекундах, в течение которого осуществляется подача), вторая команда – поворот сервопривода с камерой на заданное число градусов.

Такие пакеты данных отправляются через последовательный порт с микрокомпьютера Raspberry на плату Arduino, которая уже непосредственно осуществляет управление моторами и сервоприводом.

***Запуск и отладка***

В связи с изначальным выбором в качестве источника питания для робота обыкновенных батареек (4 штуки), команда столкнулась с проблемой нехватки питания для обоих плат и было решено использовать вдобавок к батарейкам ещё и портативное зарядное устройство power bank. На рисунке 10 представлена конфигурация робота, в которой команда представила его на соревнованиях в ЦРР, с использованием обычных батареек и power bank.

**Рис.10 *Целиком собранный робот (первый вариант)***

Идея, изложенная выше оказалась несостоятельной, т.к. мощности всё равно не хватало для платы Arduino и моторов, и команда приняла решение закупить аккумуляторы 18600, с которыми в конечном итоге система заработала как нужно.

**Рис.11** *Аккумуляторы 18600* **Рис.12** *Финальная сборка робота (окончательный вариант)*

Для запуска основного приложения, обеспечивающего работу автономного устройства, необходимо или прописать его в автозапуск операционной системы Raspberry, но в таком случае для остановки или перезапуска приложения понадобилось бы перезагружать саму Raspberry, либо организовать связь робота с ноутбуком через SSH (SSH — это [сетевой протокол,](http://fb.ru/article/148419/setevoy-protokol---eto-chto-osnovnyie-setevyie-protokolyi) с помощью которого производится безопасное управление операционной системой удаленного узла в сети. Обеспечивает защищенное соединение, аутентификацию и передачу данных с одного хоста на другой благодаря шифрованию трафика, проходящего через него. Он позволяет создавать шифрованные туннели для безопасной передачи других сетевых протоколов через незащищенную сеть, например, Интернет. Часто его используют для перенаправления портов одного компьютера на порты другого). Связь платы с ноутбуком через SSH также позволяет удалённо редактировать исходный код приложения, компилировать его заново в консоли и запускать неограниченное число раз без перезагрузки микрокомпьютера Raspberry. При этом оба компьютера подключены к общей сети Wi-Fi. Для установки соединения используется приложение Bitvise SSH Client.