Федеральное государственное бюджетное

образовательное УЧРЕЖДЕНИЕ ВЫСШЕГО Образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИМ. Н. П. ОГАРЁВА»**

Институт электроники и светотехники

Кафедра автоматизированных систем обработки информации и управления

ОТЧЁТ

по технологической практике

студента 2 курса

направление подготовки 09.04.01

код

г. Саранск, МГУ им. Н. П. Огарева, ИЭС, кафедра АСОИУ

место прохождения практики: город, предприятие,

срок прохождения практики с 11 ноября по 07 декабря 2019 года

срок прохождения практики

Автор отчёта Е. А. Немчинова

подпись, дата, инициалы, фамилия

Обозначение отчёта ОП – 02069964 –09.04.01– 6 – 19

Руководитель практики:

от университета Н. П. Плотникова

подпись, дата, инициалы, фамилия

Отчёт защищён 07 декабря 2019 года Оценка \_\_\_\_\_\_\_\_

дата

Саранск

2019

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИМ. Н. П. ОГАРЁВА»**

Институт электроники и светотехники

Кафедра автоматизированных систем обработки информации и   
управления

УТВЕРЖДАЮ

Зав. кафедрой

к.т.н, проф

С. А. Федосин

(подпись)

«\_\_\_»\_\_\_\_\_\_\_\_20\_\_г.

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

на технологическую практику

вид практики

студенту Немчиновой Елене Андреевне

фамилия, имя, отчество

2 курса, 641 группы, направление подготовки 09.04.01

код

Место прохождения практики г. Саранск, МГУ им. Н. П. Огарева, ИЭС,

кафедра АСОИУ город, предприятие, цех (отдел), организация, фирма

Срок прохождения практики с 11 ноября по 07 декабря 2019 года

начало (дата) – окончание (дата)

Срок представления отчёта и отзыва руководителя практики от профильной организации на защиту 07 декабря 2019 г.

дата

1. Цели и задачи практики

Целями прохождения технологической практики является формирование компетенций, необходимых для реализации профессиональной и научной деятельности по направлению 09.04.01 – «Информатика и вычислительная техника», а также закрепление знаний, приобретаемых обучающимися в ходе освоения учебного плана:

– получение на практике умений и навыков в организации исследовательских и проектных работ;

– освоение технологических процессов, конструктивных элементов основного и вспомогательного оборудования, методов лабораторных испытаний;

– закрепление и углубление знаний, полученных в процессе теоретического обучения, применение их на практике;

– накопление новых знаний в области вычислительной техники, информационных технологий и систем, применяемых на предприятиях и в организациях;

– изучение программного, аппаратного и информационного обеспечения управляющих систем различного уровня и назначения;

– правильное понимание существующих подходов к верификации моделей программного обеспечения.

Задачи технологической практики:

– изучение действующих стандартов, технических условий, должностных обязанностей, положений и инструкций по эксплуатации ВТ;

– изучение аппаратного и программного обеспечения, используемого на предприятии;

– изучение требований к оформлению технической документации;

– изучение правил эксплуатации средств ВТ, исследовательских установок, имеющихся на предприятии, а также их обслуживания;

– систематизация и углубление теоретических знаний и практических навыков в области применения средств разработки программно- информационных систем.

2. Компетенции студента, формируемые в результате прохождения практики:

УК-6 – Способность определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки;

ПК-1 – Управление работами по сопровождению и проектами создания (модификации) ИС, автоматизирующих задачи организационного управления и бизнес-процессы;

ПК-2 – Управление проектами в области ИТ малого и среднего уровня сложности в условиях неопределенностей, порождаемых запросами на изменения, с применением формальных инструментов управления рисками и проблемами проекта;

ПК-3 – Управление развитием БД;

ПК-4 – Управление развитием инфокоммуникационной системы организации;

ПК-5 – Интеграция разработанного системного программного обеспечения.

Индивидуальное задание на практику

Моделирования и конструирование робота-гексапода, разработка и реализация алгоритма перемещения робота.

Руководитель практики

от Университета 07.12. 2019 г. Н. П. Плотникова

подпись, дата, инициалы, фамилия

Задание к исполнению принял 07.12. 2019 г. Е. А. Немчинова

подпись, дата, инициалы, фамилия

**РЕФЕРАТ**

Отчёт по технологической практике содержит 29 страниц, 4 рисунка, 1 таблицу, 12 использованных источников.

РОБОТ-ГЕКСАПОД,АЛГОРИТМ ПЕРЕМЕЩЕНИЯ, ARDUINO, СЕРВОПРИВОД, МОДЕЛЬ.

Практика — это вид учебной работы, основным содержанием которой является выполнение практических учебных, учебно-исследовательских работ, практика по получению профессиональных умений и навыков, исполнительская практика, и другие формы по усмотрению образовательной организации творческих заданий на предприятиях, в организациях или учреждениях, соответствующих характеру будущей профессиональной деятельности.

Практика направлена на приобретение студентами умений и навыков по выбранному направлению.

Место проведения практики – г. Саранск, МГУ им. Н. П. Огарева, ИЭС, кафедра АСОИУ.

Во время практики выполнено индивидуальное задание, проделанная работа отражена в дневнике практики.

Получен положительный отзыв руководителя практики.

*Отчет по технологической практике*

*ОП – 02069964 –09.04.01 – 6 – 19*

*МГУ им. Н.П. Огарева,*

*ИЭС, АСОИУ, 641*

*Лит. Лист Листов*

*4*

*29*

*Разраб.*

*Пров.*

*Н.контр.*

*Утв.*

*Изм*

*Лист*

*№ докум.*

*Подп.*

*Дата*

*Немчинова*

*Плотникова*

принял 5.11.2018 г. Н. В. Макарова

подпись, дата, инициалы, фамилия

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| ВВЕДЕНИЕ……………………………………………………………………  1 Моделирование и конструирование робота………………………………  1.1 Описание модели робота………………………………………….  1.2 Выбору комплектующих робота-гексапода………………………  2 Разработка программ управления роботом в среде Arduino IDE…………  2.1 Среда разработки Arduino IDE ……………………………………  2.2 Написание программы для Arduino………………………………  2.3 Алгоритмы перемещения гексапода………………………………  ЗАКЛЮЧЕНИЕ………………………………………………………………  СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……..………………… | 7  8  8  10  12  12  13  14  27  28 |

# **ВВЕДЕНИЕ**

Одним из этапов обучения в магистратуре является прохождение технологической практики. Этот вид деятельности магистранта нацелен на подготовку будущих магистров к профессиональной деятельности. В ходе практики закрепляются теоретические знания, полученные в процессе обучения в университете, происходит знакомство с организацией работы и структурой предприятия, даются четкие представления о характере предстоящей трудовой деятельности. Знания, умения и навыки, сформированные в результате прохождения технологической практики, используются в дальнейшем при прохождении преддипломной практики, при выполнении научно- исследовательской работы, при подготовке магистерской диссертации.

Практика проходила на базе кафедры АСОИУ Института электроники и светотехники МГУ им. Н. П. Огарева с 11 ноября 2019 по 7 декабря 2019 года.

В результате прохождения практики была сконструирована модель робота-гексапода, подобран и реализован наиболее подходящий алгоритм движения перемещения робота, который позволит учесть особенности движения модели при дальнейшем применении искусственного интеллекта для управления им. Также в научном журнале была опубликована статья с описанием процесса моделирования и конструирования гексапода.

# **1 Моделирование и конструирование робота**

## **1.1 Описание модели робота**

В качестве модели будем использовать робота-гексапода, состоящего из туловища и шести конечностей, прикрепленных к нему (рисунок 1). Он имеет простую конструкцию, его детали легко можно найти в интернет-магазинах или сделать самостоятельно.  Внешне робот напоминает паука с шестью ножками [1].

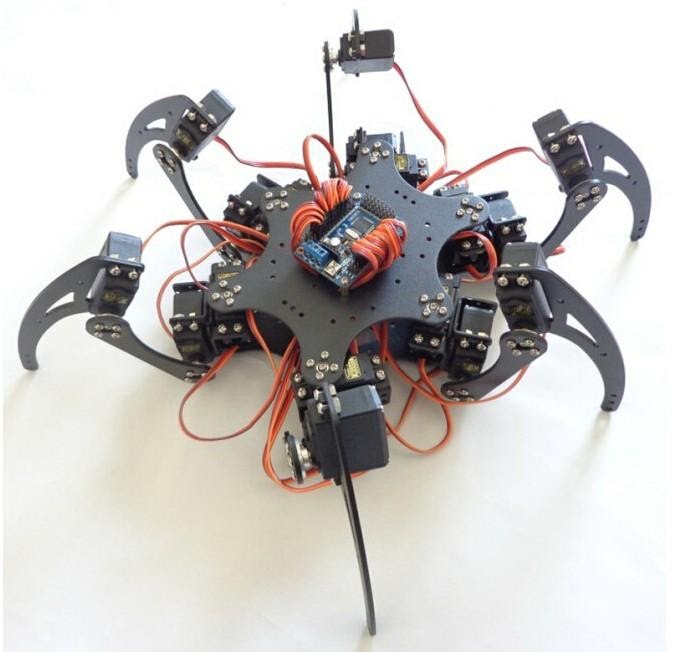


Рисунок 1 – Робот-гексапод

Тело конструируемого робота при этом состоит из двух параллельных пластин (верхней и нижней), на которых расположены отверстия и выступы для крепления платы и ног [2, 3]. Форма конечностей гексапода заимствована из биологии, поэтому для упрощения идентификации нужной части будет использоваться соответствующая терминология (рисунок 2).

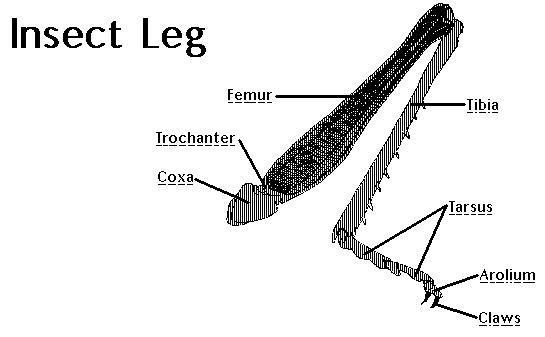


Рисунок 2 − Анатомия ноги насекомого

Конструкция конечности модели включает тазобедренный сустав (Coxa), бедро (Femur) и сустав, в котором оно совершает поворот (Trochanter), а также голень (Tibia) со своим суставом движения.  Каждый сустав в модели представляется отдельным сервоприводом.

Конечности паука имеют три степени свободы для реализации возможности более адаптированного перемещения в трехмерном пространстве. Движения совершаются в трех точках конечности [10, 11]. В суставе Coxa сервопривод поворачивает всю ногу таким образом, что любая ее точка перемещается по окружности, лежащей в плоскости, параллельной пластинам корпуса. Сустав Trochanter связан с бедром Femur и будет поднимать и опускать его. Вместе с ним, как единая система, перемещается и часть Tibia. В момент работы сустава Trochanter в теории она считается неподвижной, жестко связанной с Femur. Однако на практике можно заметить, что действия могут совершаться параллельно, и Tibia будет поворачивается в соответствующем суставе вместе с работой остальных суставов конечности. Сустав голени Tibia вращает ее, перемещая вверх и вниз [5, 6].

**1.2 Выбору комплектующих робота-гексапода**

**Материнская плата**

Основным элементом является материнская плата. В настоящее время одной из самых популярных плат для подобных задач является Arduino. Будем использовать Arduino Uno. Она выполнена на базе процессора ATmega328p с тактовой частотой 16 МГц, обладает памятью 32 КБ и имеет 20 контролируемых контактов ввода и вывода для взаимодействия с внешним миром. Arduino является открытой платформой, состоящей из аппаратной и программной частей. Для программирования используется упрощенная версия языка C++. Разработку можно вести как в бесплатной среде Arduino IDE, так и с помощью произвольного инструментария C/C++. Программирование и общение платы с компьютером осуществляется через USB-кабель [7].

**Сервоприводы**

Так как корпус гексапода металлический, то вся конструкция получается достаточно тяжелой, поэтому необходимо, чтобы сервоприводы смогли выдержать создаваемую нагрузку. Подходящим сервоприводом является TowerPro MG996R. Механизм сервопривода выполнен из металла. Диапазон вращения равен 180 градусам, что достаточно для поставленной задачи. Шестиногому гексаподу необходимо 18 сервоприводов для обеспечения работы суставов каждой из конечностей [7].

Для удобства подключения сервоприводов и их корректной работы на материнскую плату обычно устанавливается дополнительная плата расширения. В данном случае будем использовать Multiservo Shield, подходящий для многосуставных роботов. Он дает возможность управлять 18 сервоприводами. На плате расширения установлен отдельный микроконтроллер ATmega48, все силы которого направлены на то, чтобы аккуратно и точно в нужное время передавать управляющие сигналы на подключенные сервоприводы. Это позволяет избежать подергиваний приводов в произвольные моменты времени, как это происходит при использовании стандартной библиотеки Servo. В дополнение к 18 выводам для сервоприводов, управляемым выделенным микроконтроллером, на плату также вынесены 6 выводов Arduino напрямую. Таким образом, возможное количество сервоприводов в устройстве может достигать 24 штук. Шилд имеет разъемы для своего собственного питания. Если питание не подведено, то он работает от самой платы Arduino [7]. Пример подключение сервоприводов к Multiservo Shield представлен на рисунке 3.

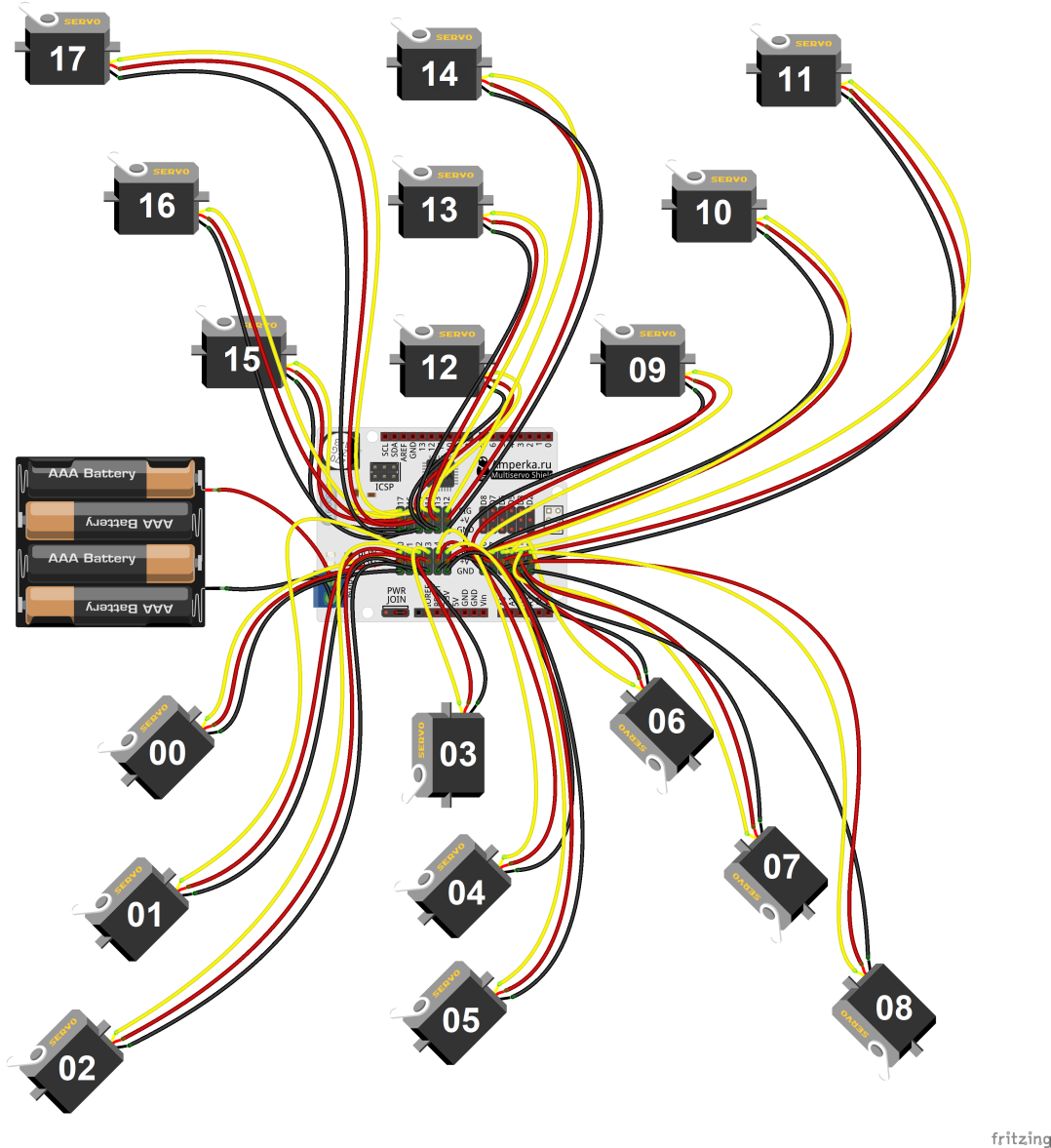


Рисунок 3 – Пример подключения сервоприводов гексапода   
к Multiservo Shield

На рисунке приведена нумерация сервоприводов в соответствии с номерами выводов платы Multiservo Shield, к которым они подключены.

**2 Разработка программ управления роботом в среде Arduino IDE**

**2.1 Среда разработки Arduino IDE**

Для написания программ по управлению конечностями гексапода будем использовать среду Arduino IDE. Это интегрированная среда разработки, которая является кросс-платформенным приложением (для ОС Windows, MacOS, Linux). Она используется для написания и загрузки программ на Arduino-совместимые платы, а также платы других производителей с помощью специальных средств [4]. Исходный код для среды IDE выпущен под общедоступной лицензией GNU версии 2. Arduino IDE поддерживает языки C и C ++ с использованием специальных правил структурирования кода. Arduino IDE предоставляет библиотеку программного обеспечения, которая предоставляет множество общих процедур ввода и вывода. Для написанного пользователем кода требуются только две базовые функции: для запуска инициализации и основного цикла программы. Arduino IDE использует программу avrdude для преобразования исполняемого кода в текстовый файл в шестнадцатеричной кодировке, который загружается на плату Arduino программой-загрузчиком в прошивке платы [9].

Среда разработки Arduino состоит из встроенного текстового редактора программного кода, области сообщений, окна вывода текста (консоли), панели инструментов с кнопками часто используемых команд и нескольких меню (рисунок 4). Для загрузки программ и связи среда разработки подключается к аппаратной части Arduino [9].

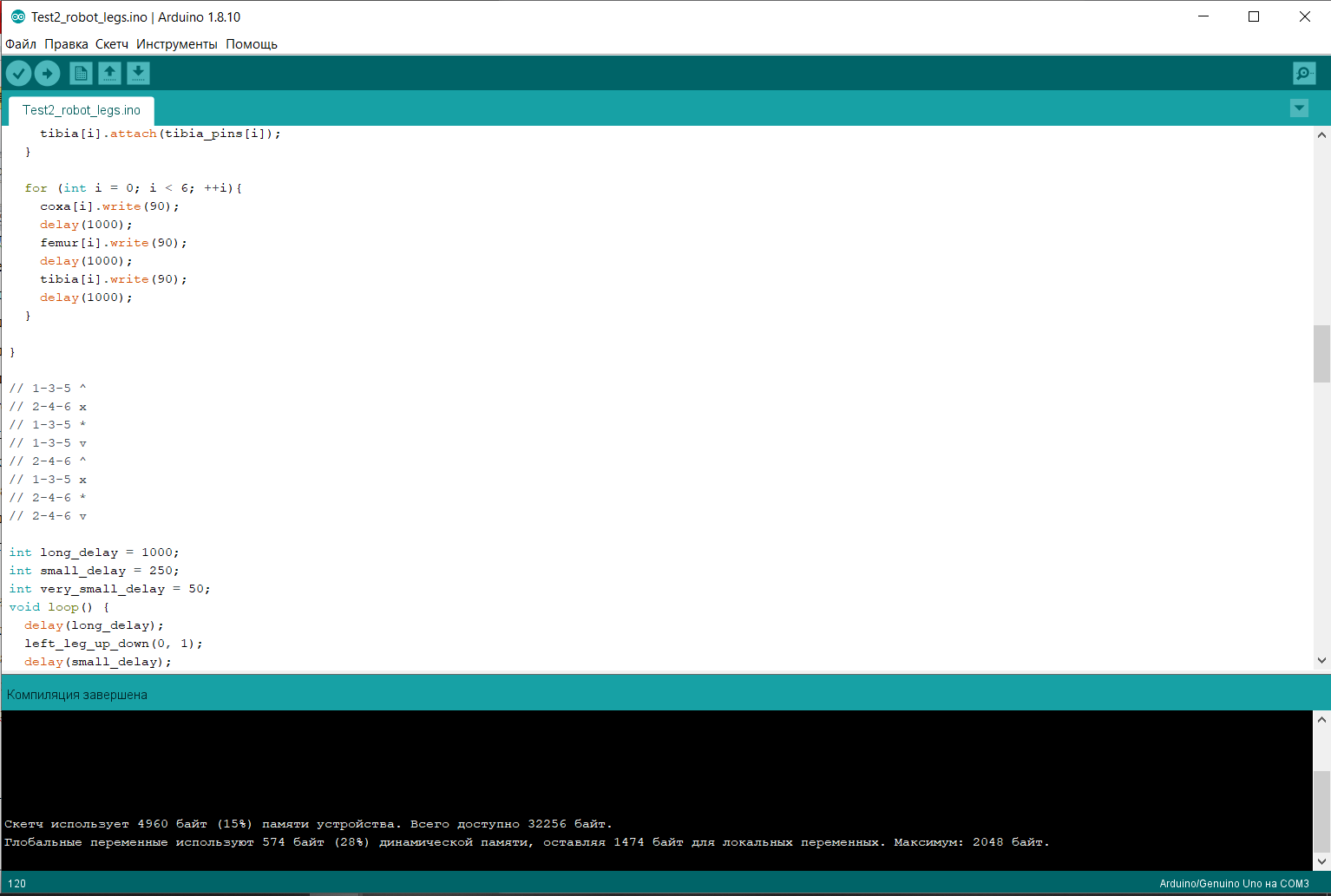


Рисунок 4 – Окно редактора среды разработки Arduino

Программа, написанная в среде Arduino, называется скетч. Скетч пишется в текстовом редакторе, имеющем инструменты вырезки/вставки, поиска/замены текста. Во время сохранения и экспорта проекта в области сообщений появляются пояснения, также могут отображаться возникшие ошибки. Окно вывода текста (консоль) показывает сообщения Arduino, включающие полные отчеты об ошибках и другую информацию. Кнопки панели инструментов позволяют проверить и записать программу, создать, открыть и сохранить скетч.

**2.2 Написание программы для Arduino**

Язык программирования устройств Arduino основан на C/C++ и скомпонован с библиотекой AVR Libc и позволяет использовать любые ее функции. Основными и обязательными функциями в структуре скетча являются setup() и loop(). Функция setup() вызывается, когда стартует скетч. Используется для инициализации переменных, определения режимов работы выводов, запуска используемых библиотек и т.д. Функция setup запускает только один раз, после каждой подачи питания или сброса платы Arduino. После выполнения функции setup() происходит циклический вызов функции loop(), которая позволяет программе постоянно совершать вычисления и реагировать на них [3].

Как и в языках C/C++ в скетчах для Arduino с помощью директивы   
#include подключаются необходимые библиотеки. Чтобы программа могла управлять сервоприводами через подключенный шилд, требуется использовать библиотеку Multiservo. Она предоставляет набор средство для удобного взаимодействия с сервоприводами робота. Первоначально создается экземпляр класса Multiservo, с помощью которого будет осуществляться соединение с конкретным сервоприводом. Затем методом attach() подключается созданный объект к указанному выходу на плате шилда, с которого осуществляется управление приводом. Функцией write() задается угол поврода сервопривода. Для считывания текущего положения (угла), в котором находится сервопривод, используется функция read().

Отладочный вывод осуществляется путем передачи сообщений по последовательному соединению. Чтобы открыть последовательный порт, необходимо вызвать функцию Serial.begin(speed), передав в нее в качестве параметра скорость передачи данных в бит/c. После этого можно выводить сообщения в специализированную консоль функцией Serial.println(msg).

**2.3 Алгоритмы перемещения гексапода**

Для того чтобы проверить работоспособность и функциональные возможности сконструированного робота, было решено написать для него алгоритмы перемещения.

Первоначально обратимся к рисунку 3 и введем следующие обозначения. Будем считать, что сервоприводы с номерами 00, 01, 02 относятся к конечности 1, причем 00 – сустав Coxa, 01 – сустав Femur и 02 – сустав Tibia, тогда, соответственно, сервоприводы 03, 04, 05 будут отнесены к конечности 2 и т.д. Подробное распределение сервоприводов по конечностям и их частям представлено в таблице 1.

Таблица 1 – Соответствие сервоприводов частям конечностей

|  |  |  |
| --- | --- | --- |
| **Номер конечности** | **Наименование сустава** | **Номер сервопривода (вывода на плате)** |
| 1 | Coxa | 00 |
| Femur | 01 |
| Tibia | 02 |
| 2 | Coxa | 03 |
| Femur | 04 |
| Tibia | 05 |
| 3 | Coxa | 06 |
| Femur | 07 |
| Tibia | 08 |
| 4 | Coxa | 09 |
| Femur | 10 |
| Tibia | 11 |
| 5 | Coxa | 12 |
| Femur | 13 |
| Tibia | 14 |
| 6 | Coxa | 15 |
| Femur | 16 |
| Tibia | 17 |

Рассмотрим первый алгоритм движения гексапода. Он состоит из следующей последовательности шагов:

1. Поднять конечности 1, 3, 5 вверх на заданный угол (сустав Femur).

2. Повернуть назад конечности 2, 4, 6 назад на заданный угол (сустав Coxa).

3. Повернуть назад конечности 1, 3, 5 вперед на заданный угол (сустав Coxa).

4. Опустить конечности 1, 3, 5 вниз на заданный угол (сустав Femur).

5. Поднять конечности 2, 4, 6 вверх на заданный угол (сустав Femur).

6. Повернуть назад конечности 1, 3, 5 назад на заданный угол (сустав Coxa).

7. Повернуть назад конечности 2, 4, 6 вперед на заданный угол (сустав Coxa).

8. Опустить конечности 2, 4, 6 вниз на заданный угол (сустав Femur).

9. Перейти к шагу 1.

Код алгоритма для Arduino приведен имеет следующий вид:

|  |
| --- |
| #include <Wire.h>  #include <Multiservo.h>  Multiservo coxa[6];  Multiservo femur[6];  Multiservo tibia[6];  int coxa\_pins[6] = {0, 3, 6, 9, 12, 15};  int femur\_pins[6] = {1, 4, 7, 10, 13, 16};  int tibia\_pins[6] = {2, 5, 8, 11, 14, 17};  void right\_leg\_rotate(int leg\_num, bool is\_forward){  int coxa\_cur\_angle = coxa[leg\_num].read();  int delta\_ang = -10;  int end\_angle = 75;  if(!is\_forward){  delta\_ang \*= -1;  end\_angle = 105;  }  coxa[leg\_num].write(end\_angle);  }  void left\_leg\_rotate(int leg\_num, bool is\_forward){  int coxa\_cur\_angle = coxa[leg\_num].read();  int delta\_ang = 10;  int end\_angle = 105;  if(!is\_forward){  delta\_ang \*= -1;  end\_angle = 75;  }  coxa[leg\_num].write(end\_angle);  }  void right\_leg\_up\_down(int leg\_num, bool is\_up){  int femur\_cur\_angle = femur[leg\_num].read();  int delta\_ang = 10;  int end\_angle = 90;  if(is\_up){  delta\_ang \*= -1;  end\_angle = 70;  }  femur[leg\_num].write(end\_angle);  }  void left\_leg\_up\_down(int leg\_num, bool is\_up){  int femur\_cur\_angle = femur[leg\_num].read();  int delta\_ang = -10;  int end\_angle = 90;  if(is\_up){  delta\_ang \*= -1;  end\_angle = 110;  }  femur[leg\_num].write(end\_angle);  }  void rotate\_to\_start\_pos\_forward(int leg\_num){  coxa[leg\_num].write(90);  }  void setup() {  Serial.begin(9600); // открываем последовательное соединение  Wire.begin();    for (int i = 0; i < 6; ++i){  coxa[i].attach(coxa\_pins[i]);  femur[i].attach(femur\_pins[i]);  tibia[i].attach(tibia\_pins[i]);  }  for (int i = 0; i < 6; ++i){  coxa[i].write(90);  delay(1000);  femur[i].write(90);  delay(1000);  tibia[i].write(90);  delay(1000);  }  }  int long\_delay = 1000;  int small\_delay = 250;  int very\_small\_delay = 50;  void loop() {  // 1-3-5 ^  // 2-4-6 x  // 1-3-5 \*  // 1-3-5 v  // 2-4-6 ^  // 1-3-5 x  // 2-4-6 \*  // 2-4-6 v  delay(1000);  // 1-3-5 ^  left\_leg\_up\_down(0, 1);  delay(10);  left\_leg\_up\_down(2, 1);  delay(10);  right\_leg\_up\_down(4, 1);    delay(1000);  // 2-4-6 x  left\_leg\_rotate(1, 0);  delay(10);  right\_leg\_rotate(3, 0);  delay(10);  right\_leg\_rotate(5, 0);  delay(1000);  // 1-3-5 \*  left\_leg\_rotate(0, 1);  delay(10);  left\_leg\_rotate(2, 1);  delay(10);  right\_leg\_rotate(4, 1);  delay(1000);  // 1-3-5 v  left\_leg\_up\_down(0, 0);  delay(10);  left\_leg\_up\_down(2, 0);  delay(10);  right\_leg\_up\_down(4, 0);  delay(1000);  // 2-4-6 ^  left\_leg\_up\_down(1, 1);  delay(10);  right\_leg\_up\_down(3, 1);  delay(10);  right\_leg\_up\_down(5, 1);  delay(1000);  // 1-3-5 x  left\_leg\_rotate(0, 0);  delay(10);  left\_leg\_rotate(2, 0);  delay(10);  right\_leg\_rotate(4, 0);  delay(1000);  // 2-4-6 \*  left\_leg\_rotate(1, 1);  delay(10);  right\_leg\_rotate(3, 1);  delay(10);  right\_leg\_rotate(5, 1);  delay(1000);  // 2-4-6 v  left\_leg\_up\_down(1, 0);  delay(10);  right\_leg\_up\_down(3, 0);  delay(10);  right\_leg\_up\_down(5, 0);  } |

Данный алгоритм переставляет одновременно 3 ноги, что плохо влияет на устойчивость робота, так как в некоторый момент он остается на трех точках опоры. Исследования показали, что подобное движение является неэффективным, так как сервоприводы не всегда могут выдержать конструкцию гексапода [12]. В связи с этим данный алгоритм был модифицирован в следующую последовательность шагов:

1. Поднять конечность 1 вверх на заданный угол (сустав Femur).

2. Повернуть вперед конечность 1 назад на заданный угол (сустав Coxa).

3. Опустить конечность 1 вниз назад на заданный угол (сустав Femur).

4. Поднять конечность 3 вверх на заданный угол (сустав Femur).

5. Повернуть вперед конечность 3 назад на заданный угол (сустав Coxa).

6. Опустить конечность 3 вниз назад на заданный угол (сустав Femur).

7. Поднять конечность 5 вверх на заданный угол (сустав Femur).

8. Повернуть вперед конечность 5 назад на заданный угол (сустав Coxa).

9. Опустить конечность 5 вниз назад на заданный угол (сустав Femur).

10. Повернуть конечности 1-6 назад на соответствующие заданные углы (сустав Coxa).

11. Поднять конечность 2 вверх на заданный угол (сустав Femur).

12. Повернуть вперед конечность 2 назад на заданный угол (сустав Coxa).

13. Опустить конечность 2 вниз назад на заданный угол (сустав Femur).

14. Поднять конечность 4 вверх на заданный угол (сустав Femur).

15. Повернуть вперед конечность 4 назад на заданный угол (сустав Coxa).

16. Опустить конечность 4 вниз назад на заданный угол (сустав Femur).

17. Поднять конечность 6 вверх на заданный угол (сустав Femur).

18. Повернуть вперед конечность 6 назад на заданный угол (сустав Coxa).

19. Опустить конечность 6 вниз назад на заданный угол (сустав Femur).

20. Повернуть конечности 1-6 назад на соответствующие заданные углы (сустав Coxa).

21. Перейти к шагу 1.

Программная реализация описанного алгоритма:

|  |
| --- |
| #include <Wire.h>  #include <Multiservo.h>  Multiservo coxa[6];  Multiservo femur[6];  Multiservo tibia[6];  int coxa\_pins[6] = {0, 3, 6, 9, 12, 15};  int femur\_pins[6] = {1, 4, 7, 10, 13, 16};  int tibia\_pins[6] = {2, 5, 8, 11, 14, 17};  void right\_leg\_rotate(int leg\_num, bool is\_forward){  int coxa\_cur\_angle = coxa[leg\_num].read();  int delta\_ang = -10;  int end\_angle = 75;  if(!is\_forward){  delta\_ang \*= -1;  end\_angle = 105;  }  coxa[leg\_num].write(end\_angle);  }  void left\_leg\_rotate(int leg\_num, bool is\_forward){  int coxa\_cur\_angle = coxa[leg\_num].read();  int delta\_ang = 10;  int end\_angle = 105;  if(!is\_forward){  delta\_ang \*= -1;  end\_angle = 75;  }  coxa[leg\_num].write(end\_angle);  }  void right\_leg\_up\_down(int leg\_num, bool is\_up){  int femur\_cur\_angle = femur[leg\_num].read();  int delta\_ang = 10;  int end\_angle = 90;  if(is\_up){  delta\_ang \*= -1;  end\_angle = 70;  }  femur[leg\_num].write(end\_angle);  }  void left\_leg\_up\_down(int leg\_num, bool is\_up){  int femur\_cur\_angle = femur[leg\_num].read();  int delta\_ang = -10;  int end\_angle = 90;  if(is\_up){  delta\_ang \*= -1;  end\_angle = 110;  }  femur[leg\_num].write(end\_angle);  }  void rotate\_to\_start\_pos\_forward(int leg\_num){  coxa[leg\_num].write(90);  }  void setup() {  Serial.begin(9600); // открываем последовательное соединение  Wire.begin();    for (int i = 0; i < 6; ++i){  coxa[i].attach(coxa\_pins[i]);  femur[i].attach(femur\_pins[i]);  tibia[i].attach(tibia\_pins[i]);  }  for (int i = 0; i < 6; ++i){  coxa[i].write(90);  delay(1000);  femur[i].write(90);  delay(1000);  tibia[i].write(90);  delay(1000);  }  }  int long\_delay = 1000;  int small\_delay = 250;  int very\_small\_delay = 50;  void loop() {  delay(long\_delay);  left\_leg\_up\_down(0, 1);  delay(small\_delay);  left\_leg\_rotate(0, 1);  delay(small\_delay);  left\_leg\_up\_down(0, 0);  delay(long\_delay);  left\_leg\_up\_down(2, 1);  delay(small\_delay);  left\_leg\_rotate(2, 1);  delay(small\_delay);  left\_leg\_up\_down(2, 0);  delay(long\_delay);  right\_leg\_up\_down(4, 1);  delay(small\_delay);  right\_leg\_rotate(4, 1);  delay(small\_delay);  right\_leg\_up\_down(4, 0);  delay(long\_delay);  //--  rotate\_to\_start\_pos\_forward(0);  delay(very\_small\_delay);  rotate\_to\_start\_pos\_forward(2);  delay(very\_small\_delay);  rotate\_to\_start\_pos\_forward(4);  delay(very\_small\_delay);    left\_leg\_rotate(1, 0);  delay(very\_small\_delay);  right\_leg\_rotate(3, 0);  delay(very\_small\_delay);  right\_leg\_rotate(5, 0);  delay(very\_small\_delay);  //--  delay(long\_delay);  left\_leg\_up\_down(1, 1);  delay(small\_delay);  left\_leg\_rotate(1, 1);  delay(small\_delay);  left\_leg\_up\_down(1, 0);  delay(long\_delay);  right\_leg\_up\_down(3, 1);  delay(small\_delay);  right\_leg\_rotate(3, 1);  delay(small\_delay);  right\_leg\_up\_down(3, 0);  delay(long\_delay);  right\_leg\_up\_down(5, 1);  delay(small\_delay);  right\_leg\_rotate(5, 1);  delay(small\_delay);  right\_leg\_up\_down(5, 0);  delay(long\_delay);  //--  rotate\_to\_start\_pos\_forward(1);  delay(very\_small\_delay);  rotate\_to\_start\_pos\_forward(3);  delay(very\_small\_delay);  rotate\_to\_start\_pos\_forward(5);  delay(very\_small\_delay);    left\_leg\_rotate(0, 0);  delay(very\_small\_delay);  left\_leg\_rotate(2, 0);  delay(very\_small\_delay);  right\_leg\_rotate(4, 0);  delay(very\_small\_delay);  } |

Данный алгоритм имеет преимущество за счет того, что робот передвигает конечности последовательно, поэтому в каждый момент времени он находится минимум на 5 точках опоры. Это способствует его устойчивости и надежному положению в пространстве.

# **ЗАКЛЮЧЕНИЕ**

В ходе практики были изучены действующие стандарты, технические условия, должностные обязанности, положения и инструкции по эксплуатации ВТ, периферийного и офисного оборудования, требования к оформлению технической документации. Освоены языки программирования для использования их в профессиональной деятельности. Приобретены практические навыки оценки технической и организационной подготовки производства и процесса разработки программного обеспечения.

Результатом практики стали конструирование робота-гексапода, разработка и реализация алгоритма его перемещения. Было принято решение использовать в качестве материнской платы Arduino Uno, что позволило использовать простые и удобные средства для управления роботом. В процессе исследования было рассмотрено два алгоритма движения конечностей гексапода. Это позволило выделить особенности полученной модели, которые в дальнейшем будут учитываться при применении искусственного интеллекта для управления перемещением робота. По процессу конструирования была написана и опубликована статья в научном журнале.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

* + - 1. Вукобратович М. Шагающие роботы и антропоморфные механизмы. — М.: Мир, 1976. — 541 с.
      2. Гаврилов А. В. Архитектура гибридной системы управления мобильного робота //Мехатроника, автоматизация, управление. – 2004. – №. 8. – С. 30-37.
      3. Программирование Ардуино [Электронный ресурс]. — Режим доступа: https://doc.arduino.ua/ru/prog/.
      4. Среда разработки Arduino [Электронный ресурс]. — Режим доступа: http://arduino.ru/Arduino\_environment.
      5. Федосин С. А., Немчинова Е. А., Плотникова Н. П. Искусственный интеллект на базе нейронной сети, реализующий перемещение модели сложного объекта в пространстве. Наукоемкие технологии. 2018. Т. 19. №7. С. 23-29.
      6. Федосин С. А., Плотникова Н. П., Немчинова Е. А., Макарова Н. В. Особенности обучения построению моделей перемещения сложных объектов, обладающих искусственным интеллектом на базе нейронной сети. Образовательные технологии и общество. 2018. Т. 21. №3. С. 290-297.
      7. Энциклопедия знаний Амперки [Электронный ресурс]. — Режим доступа: http://wiki.amperka.ru/.
      8. Arduino IDE [Электронный ресурс]. — Режим доступа: https://all-arduino.ru/arduino-ide/.
      9. Arduino IDE [Электронный ресурс]. — Режим доступа: https://en.wikipedia.org/wiki/Arduino\_IDE.
      10. DEMYSTIFYING DEEP REINFORCEMENT LEARNING [Электронный ресурс]. ― Режим доступа: http://neuro.cs.ut.ee/demystifyingdeep-reinforcementlearning/.― Загл. с экрана.
      11. Google's DeepMind AI Just Taught Itself To Walk [Электронный ресурс]. ― Режим доступа: https://www.youtube.com/watch?v=gn4nRCC9TwQ. ― Загл. С экрана.
      12. Learn to Walk (genetic algorithm & Neural Network) [Электронный ресурс]. ― Режим доступа: https://www.youtube.com/watch?v=h-89xjWpV4U. ― Загл. с экрана.