Родина и Свобода превыше всего

FreeCAD и манипулятор ArmorX

v1.0, 12.2020

Скребцов В.И., v.skrebtsov@mail.ru

Содержание

1. Введение

Целью данной статьи является раскрытие возможностей параметрической САПР с открытым исходным кодом FreeCAD лицензии LGPL (GNU Lesser General Public License) при создании виртуальной 3D модели реального робота манипулятора ArmorX. Статья может представлять интерес не только для начинающих конструкторов, но и для специалистов IT (Information Technology), работающих в области робототехники и имеющих потребности в самостоятельном создании 3D моделей робототехнических устройств, прежде всего для целей их управления.

Цель создания манипулятора ArmorX состояла в практической демонстрации возможностей его системы управления (СУ) на базе ROS (Robot Operation System). Манипулятор ArmorX имеет пять степеней свободы (тип 5-DOF). Собственно конструкция ArmorX была собрана из деталей конструктора, приобретенного на AliExpress и созданному на базе манипулятора 5-DOF, приведенному на Рис.1. В качестве исполнительных устройств были использованы сервомоторы Lewansoul LX-16A.

Базовыми принципами, определяющими технологию создания СУ ArmorX, являются:

1. Использование исключительно свободного программного обеспечения (ПО) на базе Linux как для компонентов самой СУ ArmorX, так и инструментов для их создания. Исключается любое коммерческое ПО и ПО с закрытым исходным кодом. Именно данный принцип определил применение FreeCAD.
2. ROS является основной технологической платформой СУ ArmorX. Все базовые алгоритмы управления роботами манипуляторами, такие как прямая (FK) и инверсная (IK) кинематики, планирование траекторий, решение задачи столкновений и прочее, реализованы в рамках ROS. Точнее ROS предоставляет единые механизмы (API) для доступа к данным алгоритмам.

Для использования функционала ROS, включая визуализацию управления в Rviz и создание виртуального аппаратного симулятора манипулятора в Gazebo, необходимо создать описание манипулятора в виде файла URDF (Universal Robot Description Format). URDF содержит как геометрию (длины и локальные координатные системы звеньев) и динамические параметры (массы и моменты инерции звеньев) манипулятора, так и его визуальное представление. Для создания URDF описания необходимо предварительно получить 3D модель манипулятора. В коммерческой САПР Solidworks имеется плагин, позволяющий получить URDF из 3D модели, созданной в Solidworks. FreeCAD такой плагин пока не имеет. Но возможно, что он будет там реализован. Тем более, что FreeCAD предоставляет для этого все возможности.

Итак, мы собственно подошли к созданию 3D модели ArmorX с помощью FreeCAD. Особенность состоит в том, что по сути 3D модель ArmorX должна быть получена по реальному манипулятору ArmorX (Reverse Engineering).

В дальнейшем предполагается, что читатель знаком с базовыми основами FreeCAD, включая такие верстаки (Workbench) как Sketcher, Part, PartDesign и Assembly-4. Учитывая, что автор является специалистом в области IT, а не конструктором, то предлагаемые им решения не

обязательно будут являться самыми оптимальными. Возможно, что читатель сможет предложить собственные лучшие решения.

Рис.1. Робот-манипулятор 5-DOF

2. Основные понятия Assembly-4 и FreeCAD

В этом разделе акцентируется внимание на некоторых основных понятиях верстака Assembly-4 и FreeCAD, необходимых для дальнейшего изложения. В настоящее время FreeCAD активно развивается и мы будем использовать FreeCAD версии 0.19. В «**Приложение А»** приведен краткий справочник по типам данных, функциям и выражениям FreeCAD. В качестве основного верстака для сборки модели используем Assembly-4.

2.1. Верстак Assembly-4

Assembly-4 позволяет интегрировать в едином контейнере и в единой системе координат другие объекты FreeCAD, размещая их относительно сборки (Model) и относительно друг друга. Основными типами контейнеров Assembly-4 являются App::Part и PartDesign::Body, на которые имеются ссылки App::Link как из самой сборки (Model), так и из контейнеров типа App::Part. Детали на которые имеются ссылки, могут находиться как в одном документе (файле) со сборкой (Model), так и во внешних документах (файлах).

Поскольку Model является объектом стандартного типа FreeCAD App::Part, он может быть использован инструментами FreeCAD, работающими с объектами типа App::Part. В частности, он может вставляться в другой Model для создания вложенных сборок. Model может также содержать твердотельные неделимые (solid) детали и базовые координатные (datum) объекты. Model может быть обычной деталью, узлом (вложенная сборка) и любой их комбинацией.

Детали (Part) и ссылки (Link) на детали размещаются в сборке (Model) относительно друг друга совмещением их систем координат типа PartDesign::CoordinateSystem, называемых LCS (Local Coordinate System). Совмещение LCS выполняется встроенным движком FreeCAD ExpressionEngine. Геометрии (geometry) и ограничения (constrain) деталей не используются для их размещения друг относительно друга. Это позволяет избежать множества топологических проблем.

2.2. Скелетный эскиз (Master Sketch)

Некоторые 3D CAD системы универсального назначения предлагают рабочий процесс проектирования (workflow) основанный на сборке (assembly) полностью завершенных деталей (parts), положение которых в сборке (assembly) определяются ограничениями (constraints), наложенными геометрией элементов (geometrical features) этих деталей: отверстими (holes), краями (edges), гранями (faces) и т.д. Хотя этот подход достаточно очевиден для понимания начинающими разработчиками, однако он накладывает некоторые ограничения на продвинутых пользователей, которым необходимо модифицировать детали, уже находящиеся в сборке. Например, если положение детали в сборке опирается на некоторый геометрический элемент (geometrical feature) этой детали и этот элемент позже изменяется в процессе проектирования, то системный сборщик (assembly solver) может не иметь возможности найти исходное ограничение (original constraint) поскольку геометрический элемент, на котором оно было основано исчез. И такая ситуация вызовет ошибку сборки. Это может быть весьма сложной проблемой для решения системным сборщиком (assembly solver).

Поэтому, Assembly-4 не предлагает данный подход. Вместо этого Assembly-4 дает возможность использовать принцип скелетного эскиза (master sketch), а именно в корне (root) сборки, т.е. в Assembly-4 Model создаются один или более эскизов, которые представляют “скелет” сборки. Этот скелет соответствует функциональности сборки и содержит такие элементы как точки и линии, представляющие характерные свойства сборки: оси вращения (rotational axes), фиксированные точки соединений (fixation points), оси перемещений (translation axes), направлений лучей (beam directions) и т.д. Более того, локальные системы

координат (технически они являются объектами типа PartDesign::CoordinateSystem и называются LCS в Assembly-4) располагаются в этих характерных точках. Тот же самый принцип (локальные системы координат в характерных положениях) также применяется в процессе проектирования деталей. Поэтому локальные системы координат деталей могут соответствовать локальным системам координат самой сборки, гарантируя, что деталь размещается в сборке в нужном положении.

2.3. Вложенная сборка

Assembly-4 позволяет создавать сборку сборок, поскольку не существует различий между деталями (parts) и сборками (assemblies). «**Insert External Part**» позволяет выбирать деталь, которая имеет другие детали, присоединенные к ней. Единственное различие будет для координатных систем в подключаемых сборках. Для того, чтобы быть использованной в Assembly-4, координатная система должна находится непосредственно в корне (root) контейнера Model. Это означает, что координатная система внутри подключаемой детали не может быть использована для подключения сборки к сборке более высокого уровня.

Следовательно, для повторного использования координатной системы детали в сборке, координатная система должна быть создана в корне (root) Model и размещение (Placement) этой системы координат должно ‘копироваться’ из координатной системы, которую пользователь желает использовать. Это выполняется вставкой координатной системы и использования команды «**Place LCS**», которая позволяет выбрать подключаемую деталь в сборке и одну из ее координатных систем. Две координатные системы, одна в корне Model, а другая в подключаемой детали, будут совмещаться, даже если подключаемая деталь будет изменяться. Позволяя размещение (Placement) сборки в сборке более высокого уровня с использованием ссылки (Link) на деталь. Это звучит сложнее, чем есть на самом деле.

2.4. Локальная система координат (LCS)

Локальная система координат (LCS) является ключевым понятием FreeCAD. Любая деталь, узел или сборка имеют по-крайней мере одну систему координат LCS_origin, которая создается вместе с созданием детали, узла или сборки. Фундаментом создания сборки является операция совмещения локальных систем координат собираемых деталей и/или самой сборки. LCS представляет собой структуру данных, определяющуя положение и ориентацию объекта в пространстве. Ориентация осей LCS в пространстве подчиняется правилу правой руки.

2.5. Размещение детали (Placement)

Placement является ключевым понятием по размещению детали в сборке. Данная структура данных является неотъемлемой частью детали и полностью определяет размещение детали в сборке, т.е. ее координаты и ориентацию в пространстве.

Placement может быть установлен тремя способами:

1. Вручную, прямым редактированием ее полей.
2. Инструментом «**Transform**».

3. Инструментом Assembly-4 «**Place linked part**», который использует движок FreeCAD ExpressionEngine для вычисления Placement по значению структуры данных Attachment, принадлежащий детали. При этом LCS детали и LCS сборки совмещаются. Данный подход отличается от подхода других CAD систем, в которых размещение определяется геометриями (geometry) и ограничениями (constraint) детали, и является очень мощным.

2.6. Переменные (Variables)

Model имеет возможность создавать в своем контейнере переменные (Variables). Эти переменные могут использоваться ExpressionEngine для вычисления Placement детали сборки, параметров геометрии и ограничений детали сборки. Переменные создают гибкость и дополнительную степень свободы в процессе проектирования. В качестве значений переменных могут быть не только целые числа(Int), числа плавающей арифметики (Float) и арифметичесие выражения, но и ссылки на другие переменные.

Шаги создания переменной в Assembly-4:

- Выбрать (select) Model в дереве документа.
- Вызвать диалог создания переменной (**Assembly => Add variable**)
- Выбрать тип переменной:
- Задать имя переменной.
- Установить значение переменной по умолчанию
- **«Ok»**

2.7. Деталь (Part и Body)

Процесс создания детали App:Part или PartDesign::Body состоит из следующих шагов:

- Создать новый документ FreeCAD (**Ctrl+N**)
- Выбрать верстак **Assembly-4**
- Создать в новом документе новую деталь (**Assembly => New Part** или **Assembly => New Body**) под именем «MyName».
- Сохранить данный документ в файл (**Ctrl+S**) под именем MyName. При этом в рабочем каталоге появится файл MyName.FCStd.
- Создать или импортировать элементы геометрии в контейнер Part (App:Part) или контейнер Body (PartDesign::Body).
- Создать одну или более LCS используя **«Assembly => New Coordinate System»** и поместить их на детали туда, где они будут полезны в дальнейшем. В диалоге «Attachment» выбрать ориентиры размещения и тип привязки MapMode для создаваемой LCS.
- Сохранить деталь в файл, т.к. только сохраненные детали могут быть использованы в дальнейшем.
- Закрыть документ MyName.

2.8. Сборка (Assembly)

Процесс создания сборки состоит из следующих шагов:

- Создать новый документ FreeCAD (**Ctrl+N**)
- Выбрать верстак **Assembly-4**
- Создать в новом документе модель Model (**Ctrl+M** или **Assembly => New Model**). Это создаст новый контейнер App::Part с именем Model и структурой по умолчанию.
- Сохранить данный документ в файл (**Ctrl+S**) под именем «MyName». При этом в рабочем каталоге появится файл MyName.FCStd.

- При необходимости создать новый скелетный эскиз (Master Sketch) используя инструмент «**New Sketch**» **(Assembly => New Sketch** или одноименная иконка на панели инструментов FreeCAD) и выбрать в диалоге для него плоскость. Нарисовать скелет сборки, размещая вершины и линии в его характерных точках.
- При использовании скелетного эскиза (Master Sketch) создать новые LCS сборки используя **«Assembly => New Coordinate System»** и поместить их в корректные вершины скелетного эскиза используя тип привязки MapMode.

- Открыть (**Ctrl+O**) детали, из которых будет состоять сборка MyName.
- Выбрать элемент Model в дереве документа MyName и активировать его «**Toggle active part**» (**Right-Click**).
- Последовательно включать в сборку MyName детали используя механизм ссылок (**Ctrl+L** или **Assembly => Link a part**). При этом давать ссылкам на детали уникальные узнаваемые имена. В диалоге «**Place linked Part**» разместить ссылку на деталь в сборке MyName. При этом для каждой включаемой ссылки выбрать в качестве «**Parent Part**» подключаемый элемент сборки (уже существующую деталь, ссылку или сборку) и выбрать конкретный LCS ссылки и конкретный LCS элемента. Поворотами осей и смещениями по осям LCS ссылки относительно LCS элемента добиться результата. Подтвердить размещение «**Ok**»
- Размещение ссылки в сборке можно редактировать. Для этого выбрать ссылку на импортированную деталь в дереве сборки MyName и вызвать инструмент «**Edit Placement of a Part»** (**Assembly => Edit Placement of a Part** или одноименная иконка на панели инструментов FreeCAD). В диалоге «**Place linked Part**» выбрать LCS размещаемой ссылки детали, которую будем использовать в качестве точки соединения. В качестве «**Parent Part**» выбрать деталь, к которой будем присоединять и выбрать LCS этой детали, куда будем присоединять. Нажать «**Apply»**.
- Если размещение (Placement) соответствует цели, то подтвердить размещение «**Ok**». В противном случае выбирать другие LCS ссылки и/или детали до успешного завершения размещения.
- Если деталь корректно разместилась, но имеет неверную ориентацию и/или неверное смещение, то в диалоге «**Place linked Part**» поворотами осей и/или смещениями по осям LCS ссылки относительно LCS детали добиться результата. Подтвердить размещение «**Ok**».

3. Анализ манипулятора ArmorX

Прежде всего необходимо провести узловую деталировку манипулятора ArmorX и определить его кинематическую цепь, состоящую из звеньев (Link) и сочленений (Joint). Кинематическая цепь ляжет в основу описания URDF ArmorX и реализации анимации ArmorX штатными средствами FreeCAD.

ArmorX состоит из следующих узлов и деталей:

1. Узел сервомотора (Box) — 5 шт.
    1) Сервомотор Lewansoul LX-16A (LX-16A) — 1 шт.
    2) Левый держатель сервомотора (Keeper_1) — 1 шт.
    3) Правый держатель сервомотора (Keeper_2) — 1 шт.
    4) Кронштейн сервомотора (Holder) — 1 шт.
    5) Фланец ротора сервомотора (Flange) — 1 шт.
2. База ArmorX (Base_link) — 1 шт.
    1) Опора базы (Bracket) — 2 шт.
    2) Узел сервомотора Box сочленения A1 (Box_A1) — 1 шт.
3. Первое звено ArmorX (A1_link) — 1 шт.
    1) Узел сервомотора Box сочленения A2 (Box_A2) — 1 шт.
4. Второе звено ArmorX (A2_link) — 1 шт.
    1) Кронштейн звена (Linker) — 2 шт.
5. Третье звено ArmorX (A3_link) — 1 шт.
    1) Стержень звена (CrossBar) — 1 шт.
    2) Фланец стержня звена (CrossFlange) — 2 шт.
    3) Уголок крепления сервомоторов (Corner) — 2 шт.
    4) Узел сервомотора Box сочленения A3 (Box_A3) — 1 шт.
    5) Узел сервомотора Box сочленения A4 (Box_A4) — 1 шт.
6. Четвертое звено ArmorX (A4_link) — 1 шт.
    1) Кронштейн звена (Linker) — 1 шт.
    2) Узел сервомотора Box сочленения A4 (Box_A5) — 1 шт.
7. Манипулятор ArmorX — 1 шт.
    1) Узел Base_link — 1 шт.
    2) Узел A1_link — 1 шт.
    3) Узел A2_link — 1 шт.
    4) Узел A3_link — 1 шт.
    5) Узел A4_link — 1 шт.

Таким образом, манипулятор ArmorX состоит из следующих связанных сборок (assembly):

1. Сборку первого (верхнего) уровня составляет собственно ArmorX.
2. Сборки второго уровня составляют узлы Base_link, A1_link, A2_link, A3_link, A4_link.
3. Сборку третьего уровня составляет узел сервомотора Box.

Итак, общая подетальная спецификация ArmorX:

1. Сервомотор Lewansoul LX-16A (LX-16A) — 5 шт.
2. Левый держатель сервомотора (Keeper_1) — 5 шт.
3. Правый держатель сервомотора (Keeper_2) — 5 шт.
4. Кронштейн сервомотора (Holder) — 5 шт.
5. Фланец ротора сервомотора (Flange) - 5 шт.
6. Опора базы (Bracket) — 2 шт.
7. Кронштейн звена (Linker) — 3 шт.
8. Стержень звена (CrossBar) — 1 шт.
9. Фланец стержня звена (CrossFlange) — 2 шт.
10. Уголок крепления сервомоторов (Corner) — 2 шт.

Кинематическую цепь ArmorX составляют сборки второго уровня:
Base_link => A1_link => A2_link => A3_link => A4_link

Сочленения определяют степени свободы манипулятора ArmorX (5-DOF) и находятся в сборках второго уровня:
A1_joint (Base_link) => A2_joint (A1_link) => A3_joint (A3_link) => A4_joint (A3_link) => A5_joint (A4_link)

4. Проектирование узла сервомотора Box

Создать рабочий каталог ArmorX, в котором будут размещаться компоненты проекта ArmorX.

4.1. Сервомотор Lewansoul LX-16A (LX-16A)

3D модель сервомотора Lewansoul LX-16A формата STP была взята по ссылке https://grabcad.com/library/lewansoul-lx-16a-bus-servo-1 из свободной библиотеки 3D моделей. Поместить файл LX-16A.stp в рабочий каталог. Для использования этой модели в Assembly-4 FreeCAD потребуются некоторые дополнительные шаги:

1. Создать новый документ FreeCAD (**Ctrl+N**)
2. Выбрать верстак **Assembly-4**
3. Создать в новом документе новую деталь (**Assembly => New part**) под именем LX-16A
4. Сохранить данный документ в файл (**Ctrl+S**) под именем LX-16A. При этом в рабочем каталоге появится файл LX-16A.FCStd.

5. Импортировать файл LX-16A.stp (**Ctrl+I**). В результате в дереве документа появится контейнер Document. Изображение модели сервомотора появится в окне FreeCAD.

6. Переместить (Drag&Drop) контейнер Document в контейнер детали (part) LX-16A. Согласиться с размещением модели по умолчанию, т. е. LCS_0 сервомотора совпадает с системой координат сцены FreeCAD.

7. Создать LCS_Rotor, который будет размещаться в центре отверстия ротора сервомотора с осью Z вдоль оси ротора:
   1) Выбрать контейнер детали (part) LX-16A
   2) Создать новый LCS (**Assembly => New Coordinate System**) под именем LCS_Rotor.
   3) Разместить LCS_Rotor детали, выбрав в диалоге Attachment ориентир Part_Feature:Edge754 модели сервомотора и тип ориентирования (MapMode) Concentric. При этом ось Z ориентируется вдоль оси ротора. Установить в диалоге Attachment флажок «**Flip sides**», который сориентирует ось Z во вне от сервомотора вдоль оси ротора.

8. Для большей реалистичности модели скорректируем цвет ее элементов: цвет корпуса асфальтовый (rgb=50,50,50), цвет разъемов белый (rgb=255,255,255), цвет контактов разъемов и цвет ротора золотой (rgb=255,170,0). Для этого выберем элемент Document и вызовем (**Right-Click**) диалог «**Set colors ...**». Включить режим «**Box selection**». Сформировать группы граней модели (**Ctrl+Left-Click**) и установить нужный цвет каждой группы.

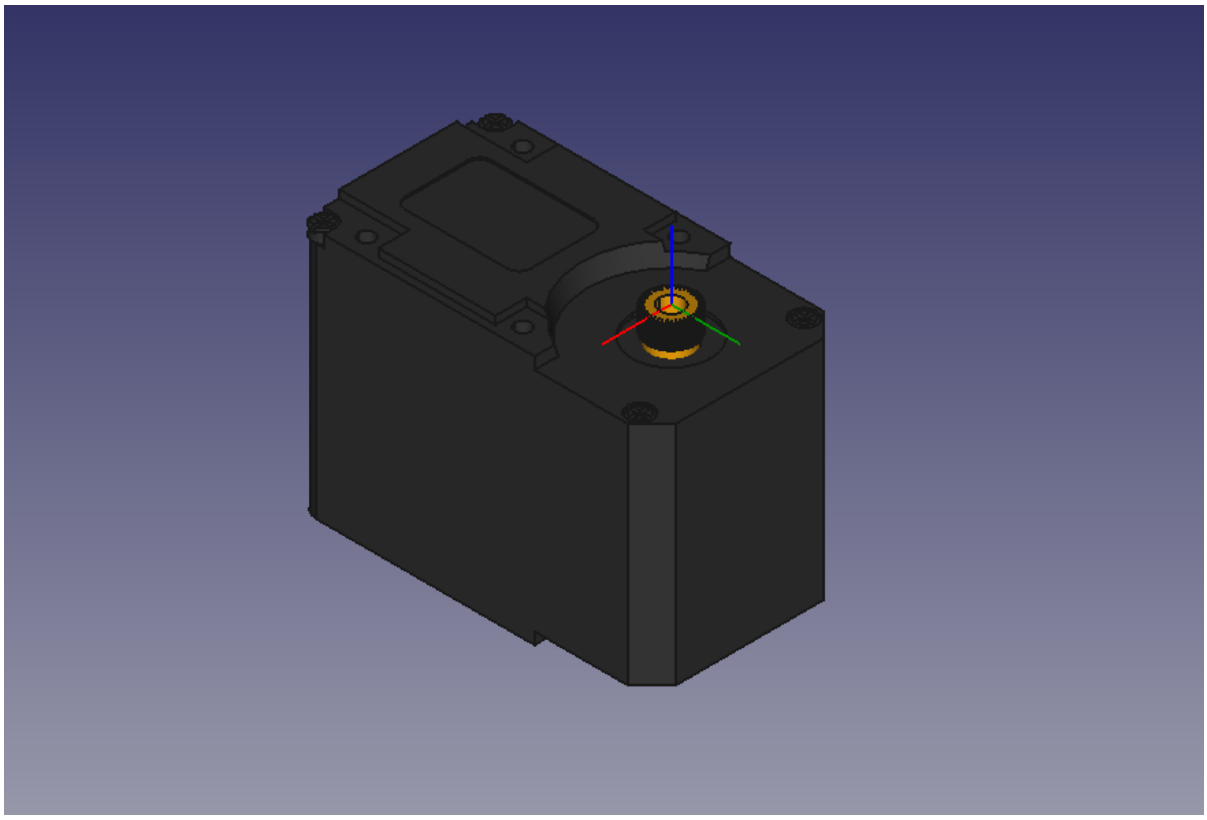

Рис. 1. Сервомотор Lewansoul LX-16A

9. Сохранить данный документ (**Ctrl+S**) и закрыть его.

4.2. Держатель сервомотора (Keeper_1 и Keeper_2)

Сервомотор LX-16A имеет два установочных места по бокам корпуса. Для проектирования левого и правого держателей сервомотора, вооружившись линейкой и штангель-циркулем, предварительно снимем размеры этих установочных мест. Размеры можно снять и по модели

сервомотора инструментом FreeCAD «**Measure distance**» (**Tools => Measure distance**). Проектирование держателя рассмотрим достаточно подробно. В дальнейшем, при проектировании других деталей будем ограничиваться лишь описанием их особенностей.

1. Создать новый документ FreeCAD (**Ctrl+N**)
2. Выбрать верстак **Assembly-4**
3. Создать в новом документе новую деталь (**Assembly => New Body**) под именем Keeper_1.
4. Сохранить данный документ в файл (**Ctrl+S**) под именем Keeper_1. При этом в рабочем каталоге появится файл Keeper_1.FCStd.

5. Выбрать элемент Keeper_1 в дереве документа и активировать его «**Toggle active body**» (**Right-Click**). При этом включается верстак **PartDesign**.

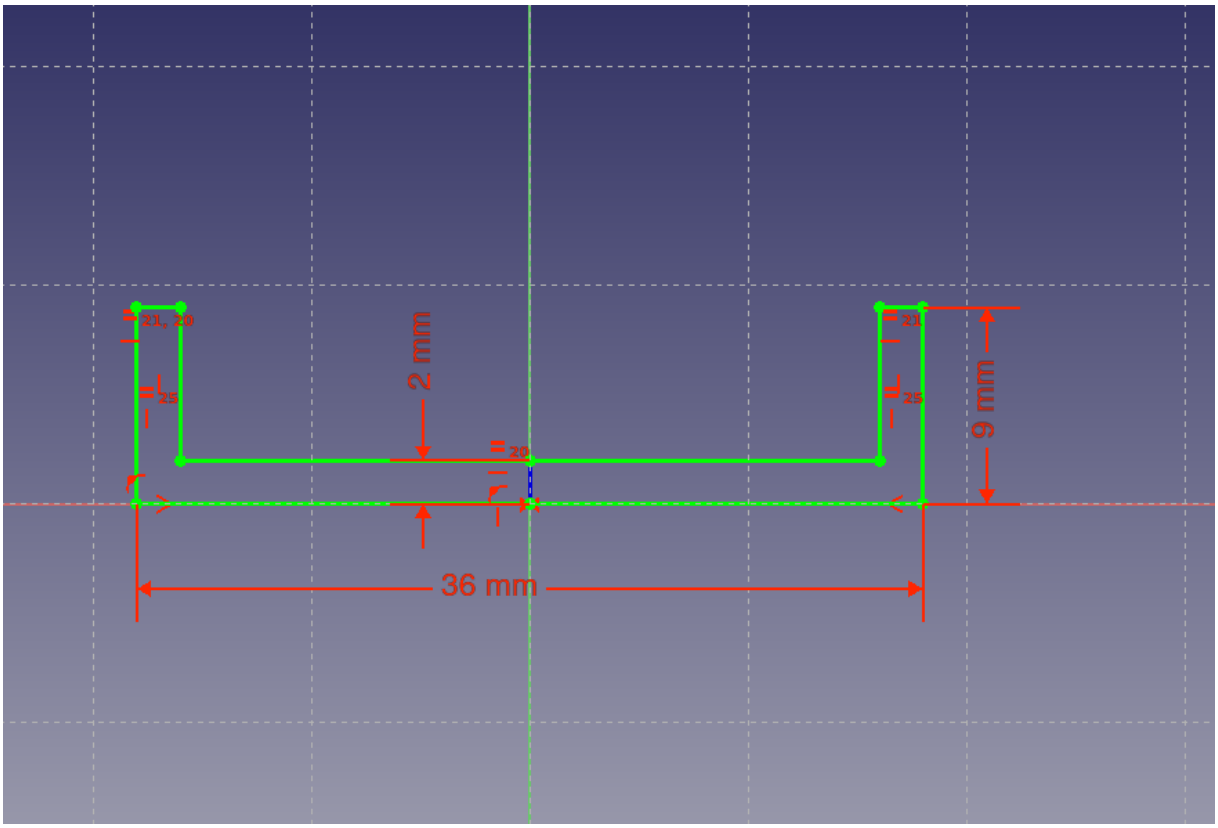

Рис. 2. Эскиз профиля держателя

6. Создать эскиз (**Sketch => Create sketch**) профиля держателя в плоскости XY и установить его размеры через систему ограничений «**Constraint**» по аналогии с Рис.2. Пожалуй единственным интересным моментом при этом является установка толщины профиля держателя. Для этого используем вспомогательный отрезок (синего цвета), длина которого равна толщине профиля 2mm, а верхние отрезки, определяющие толщину стенок держателя, устанавливаются равными вспомогательному отрезку. Переключение редактора эскиза в режим вспомогательной геометрии и обратно выполняется по **Sketch => Sketch geometries => Toggle construction geometry** или

через иконку на панели инструментов «**Toggles the toolbar to/from construction mode**».

7. Применяем операцию «выдувания» (**Pad**) к данному эскизу вдоль оси Z, задав в диалоге «**Pad parameters**» размер «выдувания» (length: 30mm) и симметрию относительно начала координат (Symmetric to plane). На Рис.3 можем видеть результат данной операции в виде заготовки держателя.

8. Приступаем к обработке заготовки держателя, в результате которой должен появиться сам держатель. Обработаем одну из сторон держателя, сделав пару отверстий и пару вырезов. Для этого выберем грань стороны держателя (**Left-Click**) и создадим на этой грани эскиз «**Create Sketch**» по аналогии с Рис.4. Для данного эскиза потребуется использовать функцию внешней геометрии **Sketch => Sketch geometries => External geometry**, которая позволяет получить в эскизе ребра боковой грани заготовки (сиреневый цвет) в качестве вспомогательных элементов. Все контуры эскиза (отверстия и вырезки) должны быть замкнутыми для дальнейшего применения к ним операции «вырезки» (**Pocket**).

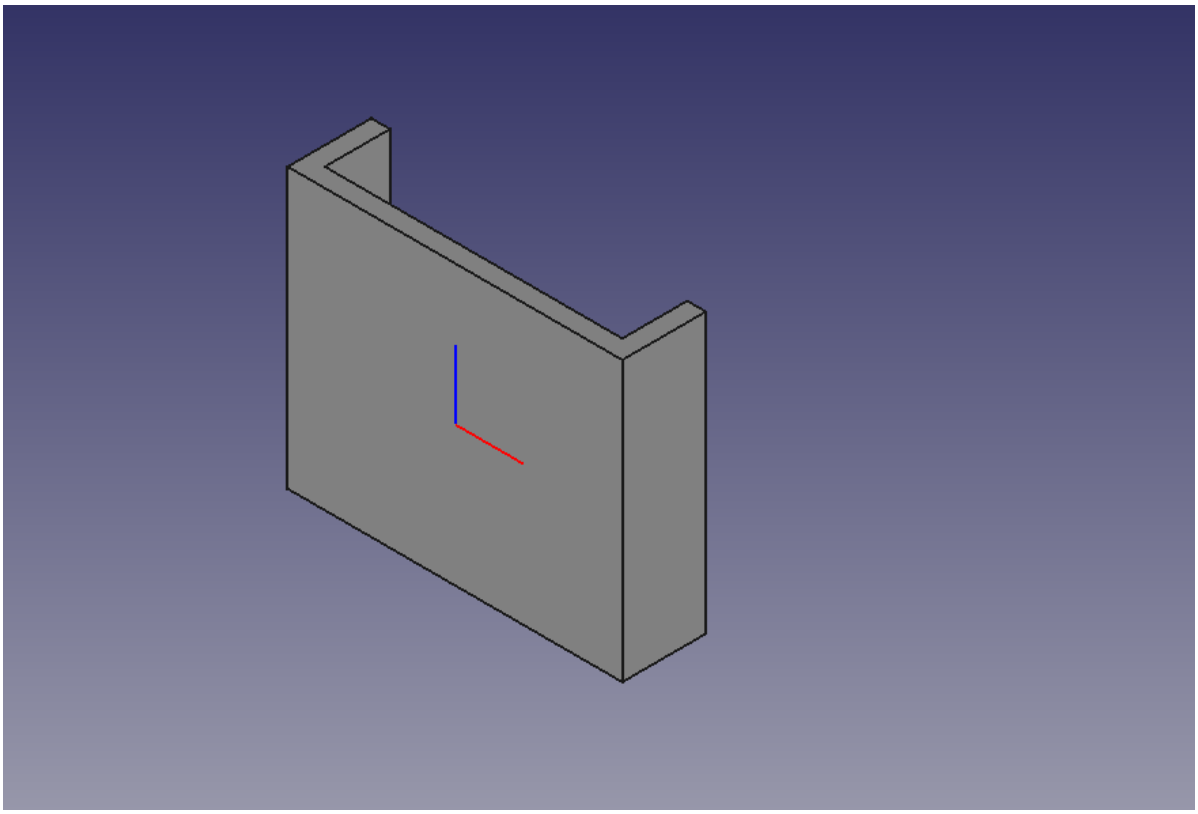

Рис. 3. Заготовка держателя

9. Применяем операцию «вырезки» (**Pocket**) к данному эскизу, задав в диалоге «**Pocket parameters**» глубину вырезки (length: 2mm). В результате получаем готовую боковую сторону держателя.

10. Вторая боковая сторона держателя имеет чуть больший вырез, чем первая. Но сам процесс ее обработки абсолютно идентичен первой стороне. Поэтому останавливаться на нем не будем.

11. В центре держателя находится одно большое отверстие, по краям которого симметрично расположены четыре маленьких отверстия. Выберем центральную грань держателя и создадим на этой грани эскиз «**Create Sketch**» по аналогии с Рис.5. В данном случае внешнюю геометрию подтягивать нет необходимости. Для симметричного размещения контуров маленьких отверстий используется вспомогательный квадрат (синего цвета) с центром в начале координат. Контуры отверстий находятся в углах данного квадрата.

12. Применяем операцию «вырезки» (**Pocket**) к данному эскизу, задав в диалоге «**Pocket parameters**» глубину вырезки (length: 2mm). В результате получаем готовую центральную часть держателя.

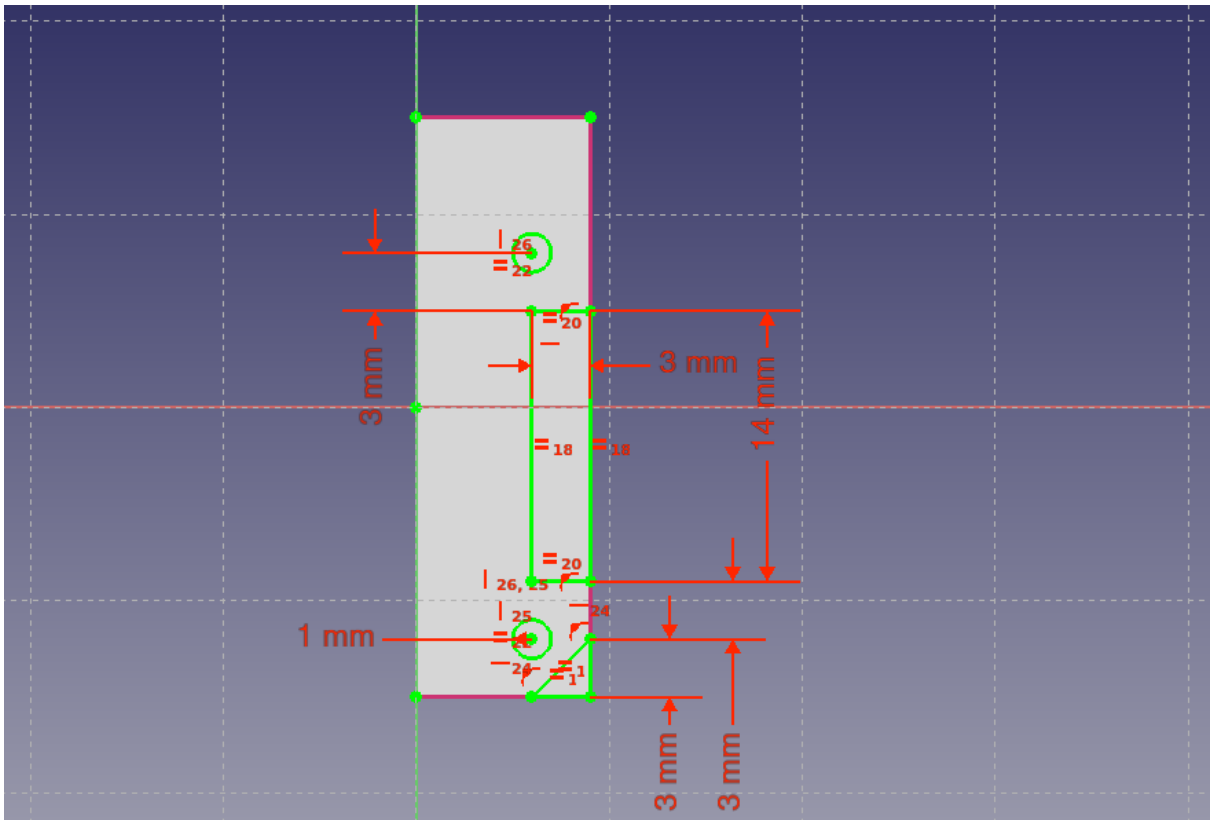

Рис. 4. Эскиз на боковой грани держателя

13. Сглаживаем внешние и внутренние углы держателя с помощью операции **Part Design => Apply a dress-up feature => Fillet** или иконки «**Fillet**» на панели инструментов FreeCAD. Выделение однотипных углов в группу **Ctrl+Left-Click**.

14. В результате получаем готовую модель держателя сервомотора Keeper_1, приведенную на Рис.6.

15. Сохранить документ Keeper_1 (**Ctrl+S**) и закрыть его.

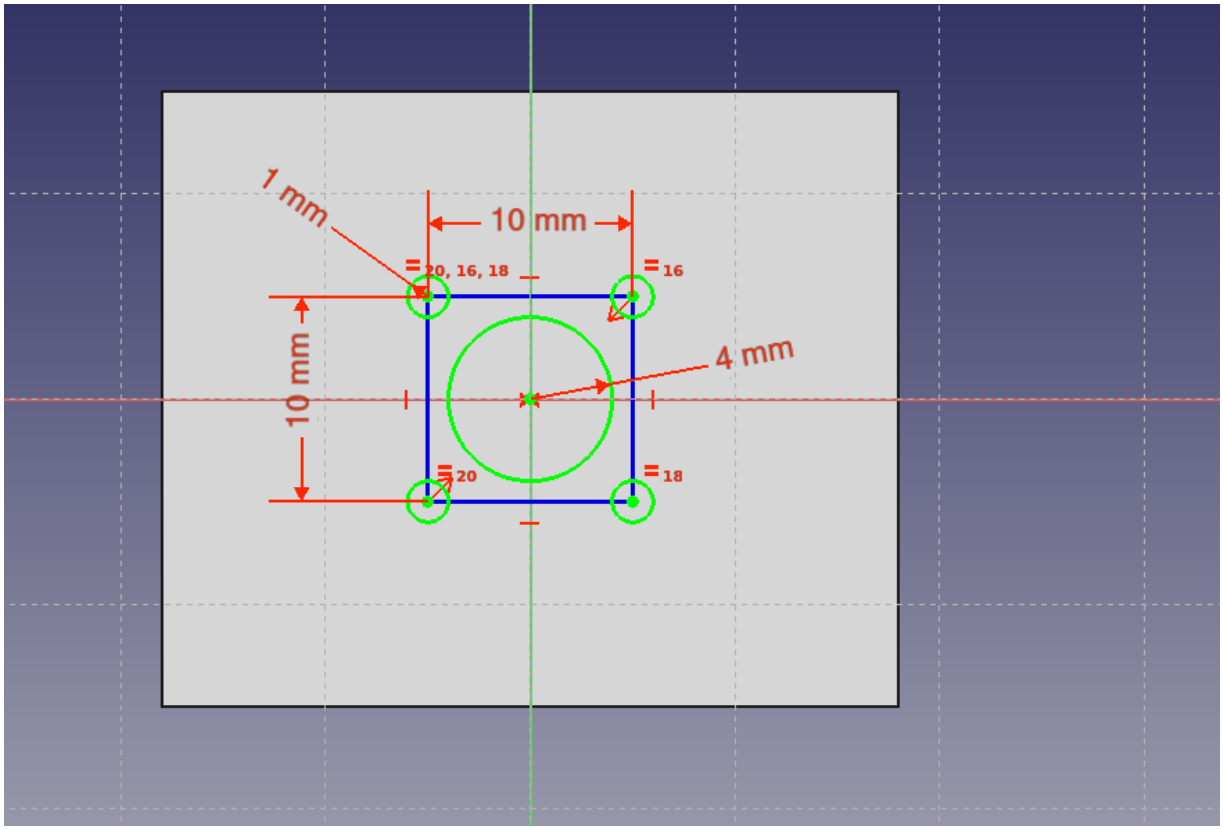

Рис. 5. Эскиз центральной грани держателя

Держатель Keeper_2 является родным братом Keeper_1 и получается из него зеркальным отражением относительно координатной плоскости XZ.

Создание держателя Keeper_2 по шагам:

1. Открыть Keeper_1.
2. Сохранить его в файл под именем Keeper_2 (**Ctrl+Shift+S**).
3. Выбрать верстак **Part**.
4. Выбрать элемент (Body) Keeper_1.

5. Зеркальное отражение детали Keeper_1 выполнить с помощью операции **Part => Mirroring …** или иконкой «**Mirroring a selected shape**» на панели инструментов FeeCAD. При этом в диалоге «**Mirroring**» выбрать в качестве исходного объекта зеркалирования (Shapes) Keeper_1, а в качестве зеркала (Mirror plane) выбрать координатную плоскость XZ. Выполнить операцию «**Ok**».

6. В окне FreeCAD появится зеркальное изображение держателя рядом с исходным. В дереве документа Keeper_2#Keeper_1 (Mirror#1).Keeper_1, выбрать его последний элемент Pocket и отключить его визуализацию (**Space**). В результате получаем только зеркальное изображение держателя.

7. Однако, для того чтобы использовать полученный держатель в сборке Assembly-4 необходимо встроить его в элемент Body с именем Keeper_2 по адресу Keeper_2#Keeper_2. Поэтому при включенном верстаке Assembly-4 создаем второй элемент Body (**Assembly => New Body**) под именем Keeper_2 и перемещаем элемент Keeper_1 (Mirror#1) в контейнер Keeper_2 (**Drag&Drop**). В результате в дереве появляется новый элемент Keeper2#Kepeer_2.Base_feature. Для единообразия сразу же переименуем в этом контейнере LCS_0001 в LCS_1.

8. Сохраняем текущий документ (**Ctrl+S**) и закрываем его.

Итак, в результате описанных действий имеем модели двух держателей сервомотора: правого и левого в файлах Keeper_1.FCStd и Keeper_2.FCStd. Модели полностью зеркально идентичны. Модель держателя сервомотора Keeper_1 представлена на Рис.6.

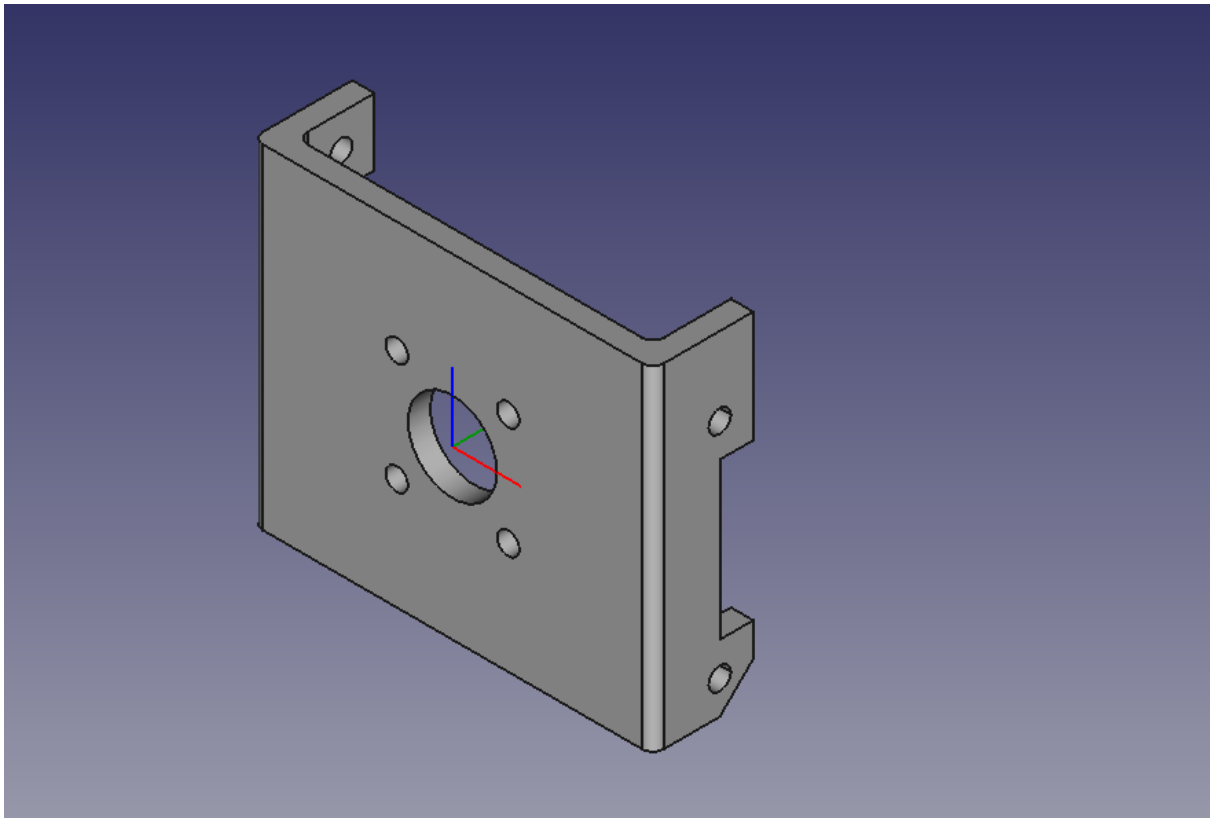

Рис. 6. Держатель сервомотора

4.3. Фланец сервомотора (Flange)

Сервомотор передает свой крутящий момент через фланец, который крепится непосредственно на ротор сервомотора.

Создание фланцы сервомотора Flange по шагам:

1. Создать новый документ FreeCAD (**Ctrl+N**)
2. Выбрать верстак **Assembly-4**

3. Создать в новом документе новую деталь (**Assembly => New Body**) под именем Flange.
4. Сохранить данный документ в файл (**Ctrl+S**) под именем Flange. При этом в рабочем каталоге появится файл Flange.FCStd.

5. Выбрать элемент Flange в дереве документа и активировать его «**Toggle active body**» (**Right-Click**). При этом включается верстак **PartDesign**.

6. Создать эскиз (**Sketch => Create sketch**) профиля фланца в плоскости XY и установить его размеры через систему ограничений «**Constraint**» по аналогии с Рис.7.
7. Применяем операцию «вращения» (**Revolution**) к данному эскизу вдоль оси X, задав в диалоге «**Revolution parameters**» ось «вращения» (Axis: Horizontal sketch axis) и угол вращения на полный оборот (Angle: 360). В результате получаем заготовку фланца.

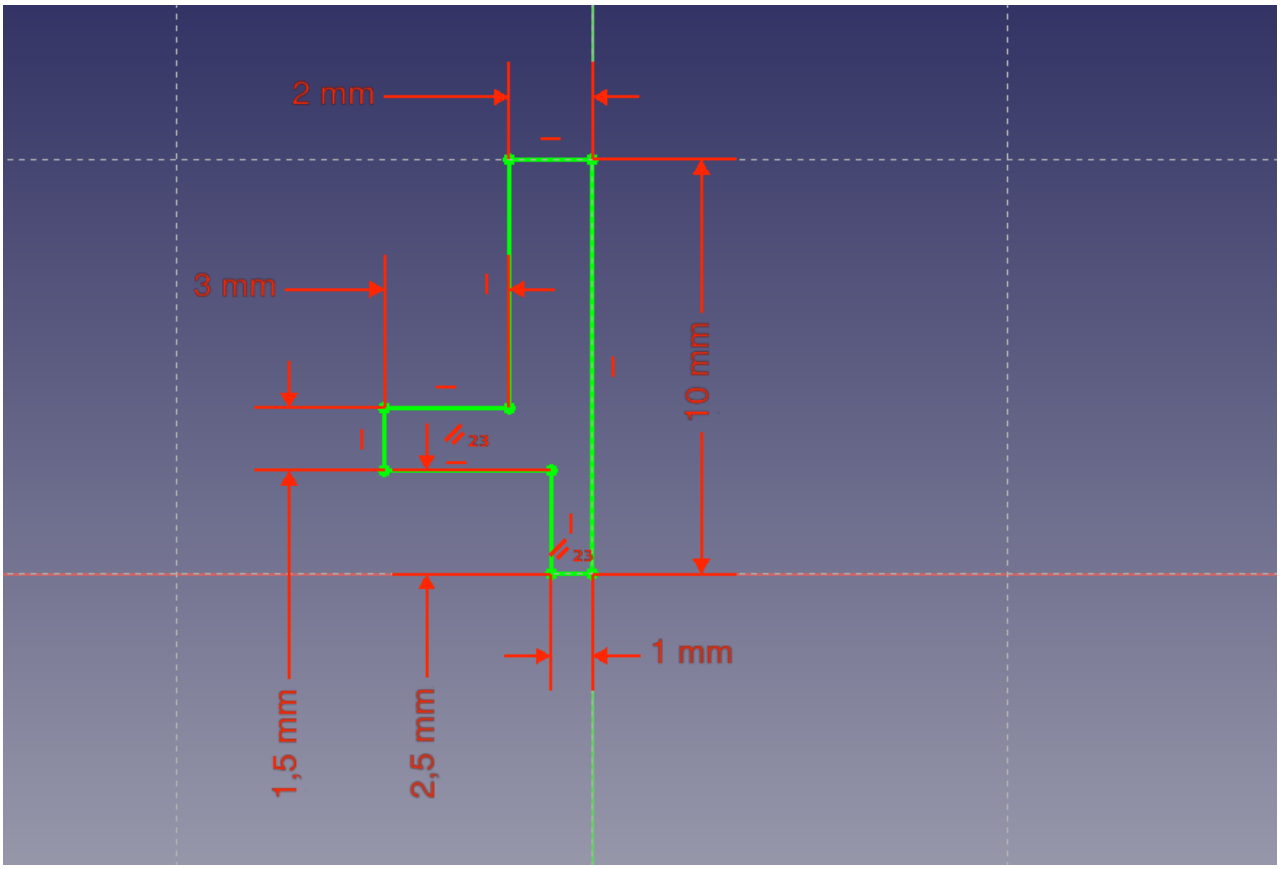

Рис. 7. Эскиз профиля фланца

8. Начинаем оформлять внешнюю грань фланца, на которой по центру находится отверстие для винта крепления фланца к ротору сервомотора. По окружности от центра симметрично расположены четыре одинаковых отверстия для крепления фланца к звеньям манипулятора. Выбираем внешнюю грань фланца (**Left-Click**) и создаем на этой грани эскиз «**Create Sketch**» по аналогии с Рис.8. Для симметричного размещения контуров четырех отверстий используется вспомогательная окружность (синего цвета) с центром в начале координат.

9. Применяем операцию «вырезки» (**Pocket**) к данному эскизу, задав в диалоге «**Pocket parameters**» глубину вырезки (length: 2mm). В результате получаем готовую внешнюю грань фланца.

10. Ротор сервомотора заканчивается шестеренкой на 40 зубьев. Поэтому мы должны создать ответную часть на внутренней стороне фланца. Для этого выбираем крайнюю грань внутренней части фланца и создаем на этой грани эскиз «**Create Sketch**» по аналогии с Рис.9. Для данного эскиза потребуется использовать функцию внешней геометрии **Sketch => Sketch geometries => External geometry**, которая позволяет получить в эскизе контур центрального внутреннего отверстия заготовки (сиреневый цвет) в качестве вспомогательного элемента. Заметим, что контур ниши для зуба должен быть замкнутым для применения операции «вырезки» (**Pocket**).

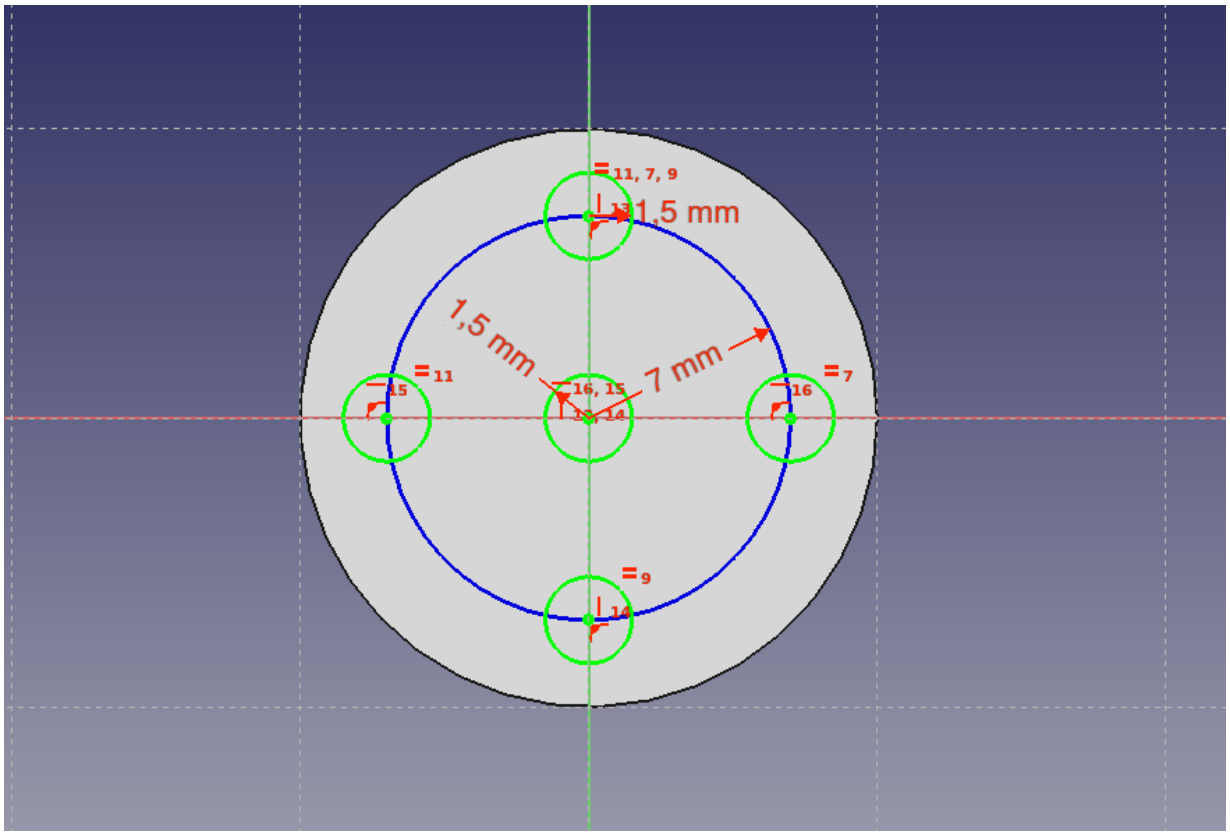

Рис. 8. Эскиз внешней грани фланца

11. Применяем операцию «вырезки» (**Pocket**) к данному эскизу, задав в диалоге «**Pocket parameters**» глубину вырезки на всю глубину центрального отверстия (length: 4mm). В результате вырезаем внутри этого отверстия нишу для одного зуба ротора.

12. Размножим эту нишу по окружности центрального отверстия с помощью операции **Part Design => Apply a pattern => Polar Pattern** или иконки «**Polar Pattern**» на панели инструментов FreeCAD. Для этого выберем последний элемент Pocket в дереве документа, зададим в диалоге полный оборт (Angle: 360) и количество зубьев (Occurrences: 40).

13. Сглаживаем внешние и внутренние углы держателя с помощью операции **Part Design => Apply a dress-up feature => Fillet** или иконки «**Fillet**» на панели инструментов FreeCAD.

14. В результате получаем готовую модель фланца сервомотора Flange, приведенную на Рис.10. Ось вращения фланца по оси X его LCS_0.

15. Сохранить документ Flange (**Ctrl+S**) и закрыть его.

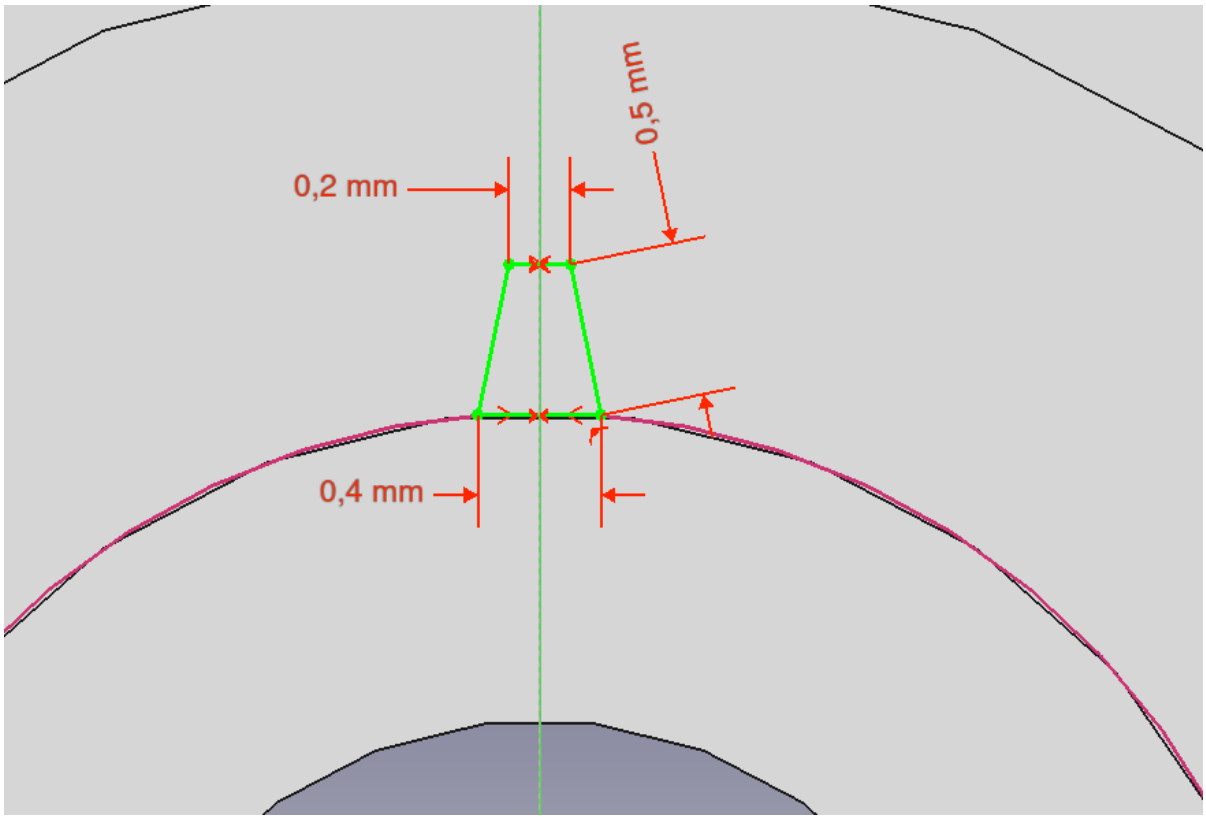

Рис. 9. Эскиз ниши для зуба зацепления ротора

4.4. Узел сервомотора Box

Для сборки узла сервомотора осталось создать кронштейн (Holder). Его создание аналогично созданию держателя (Keeper_1). Поэтому без подробных объяснений процесса его проектирования приведем лишь конечный результат на Рис.11.

Проектирование сборки узла сервомотора (Box) по шагам:

1. Создать новый документ FreeCAD (**Ctrl+N**)
2. Выбрать верстак **Assembly-4**
3. Создать в новом документе модель Model (**Ctrl+M** или **Assembly => New Model**).
4. Сохранить данный документ в файл (**Ctrl+S**) под именем Box. При этом в рабочем каталоге появится файл Box.FCStd.
5. Открыть (**Ctrl+O**) детали, из которых будет состоять сборка Box: сервомотор LX-16A, держатели Keeper_1 и Keeper_2, фланец Flange и кронштейн Holder.

6. Выбрать элемент Model в дереве документа и активировать его «**Toggle active part**» (**Right-Click**).

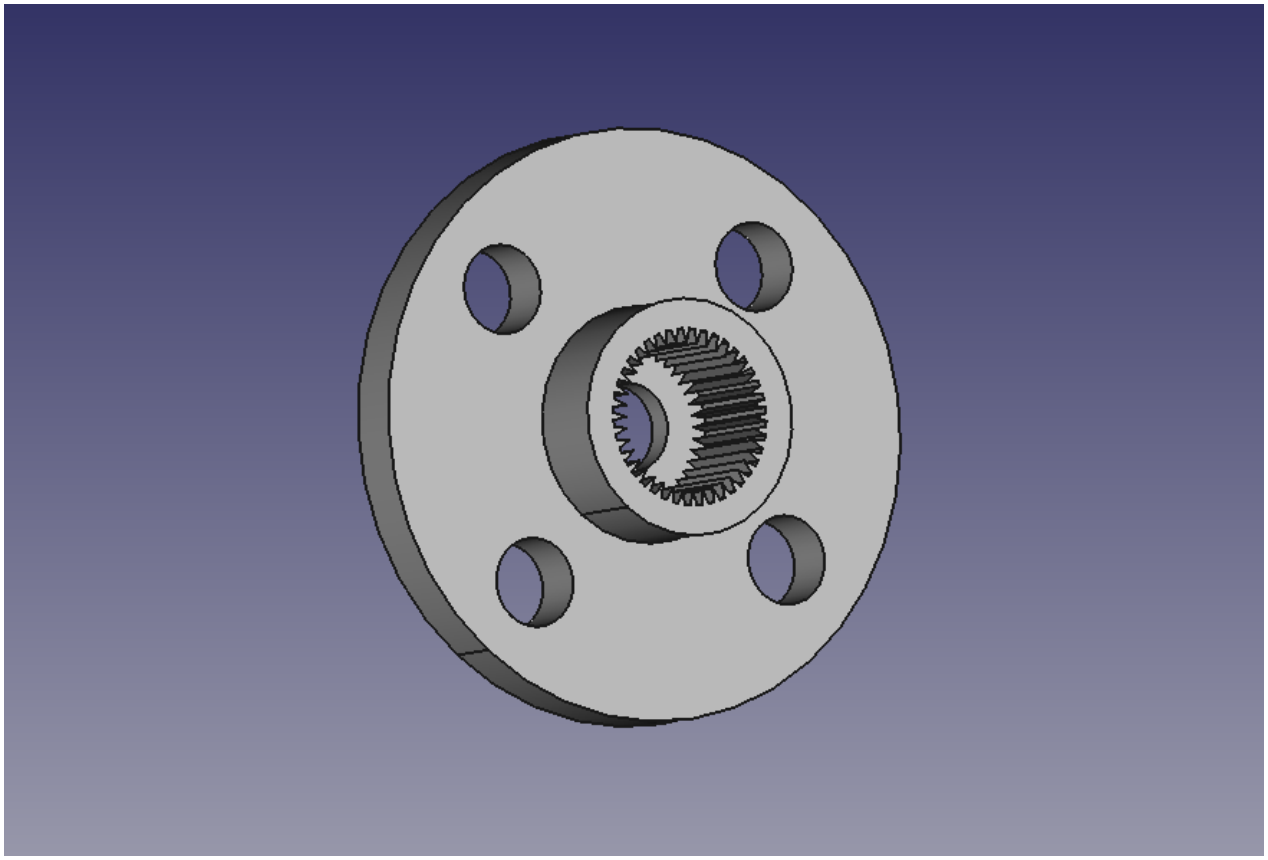

Рис.10. Фланец сервомотора

7. Включить в сборку Box сервомотор LX-16A. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем LX-16A и в диалоге «**Place linked Part**» разместить LX-16A в сборке Box так, что ось вращения ротора совпадет с осью Z базовой системы координат сборки Box. Для размещения LX-16A выбрать в качестве «**Parent Part**» Parent Assembly. Поворотами осей и смещениями по осям LCS_0 (LX-16A) относительно LCS_Origin (Box) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_0 (LX-16A) относительно LCS_Origin (Box) имеют следующие значения: X:9.5, Y:-35.0, Z:-32.0.

8. Включить в сборку Box держатель Keeper_1. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Keeper_1 и в диалоге «**Place linked Part**» разместить Keeper_1 в сборке Box так, чтобы держатель оказался сбоку сервомотора и их монтажные отверстия совпали. Поэтому для Keeper_1 в качестве «**Parent Part**» выбрать LX-16A. Если Keeper_1 не будет виден в окне FreeCAD, то в дереве документа Box#Model.Keeper_1 включить его визуализацию на последнем элементе Pocket (**Space**). Поворотами осей и смещениями по осям LCS_0 (Keeper_1) относительно LCS_0 (LX-16A) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_0 (Keeper_1) относительно LCS_0 (LX-16A) имеют следующие значения: X:14.0, Y:-6.5, Z:14.5. Точность до десятых и сотых

можно добиться прямым редактированием структуры Attachment в дереве сборки Box#Model.Keeper_1.

9. Включить в сборку Box держатель Keeper_2 по аналогии с держателем Keeper_1, но с противоположной стороны сервомотора. В качестве подсказки, смещения LCS_1 (Keeper_2) относительно LCS_0 (LX-16A) имеют следующие значения: X:14.0, Y:25.5, Z:14.5.

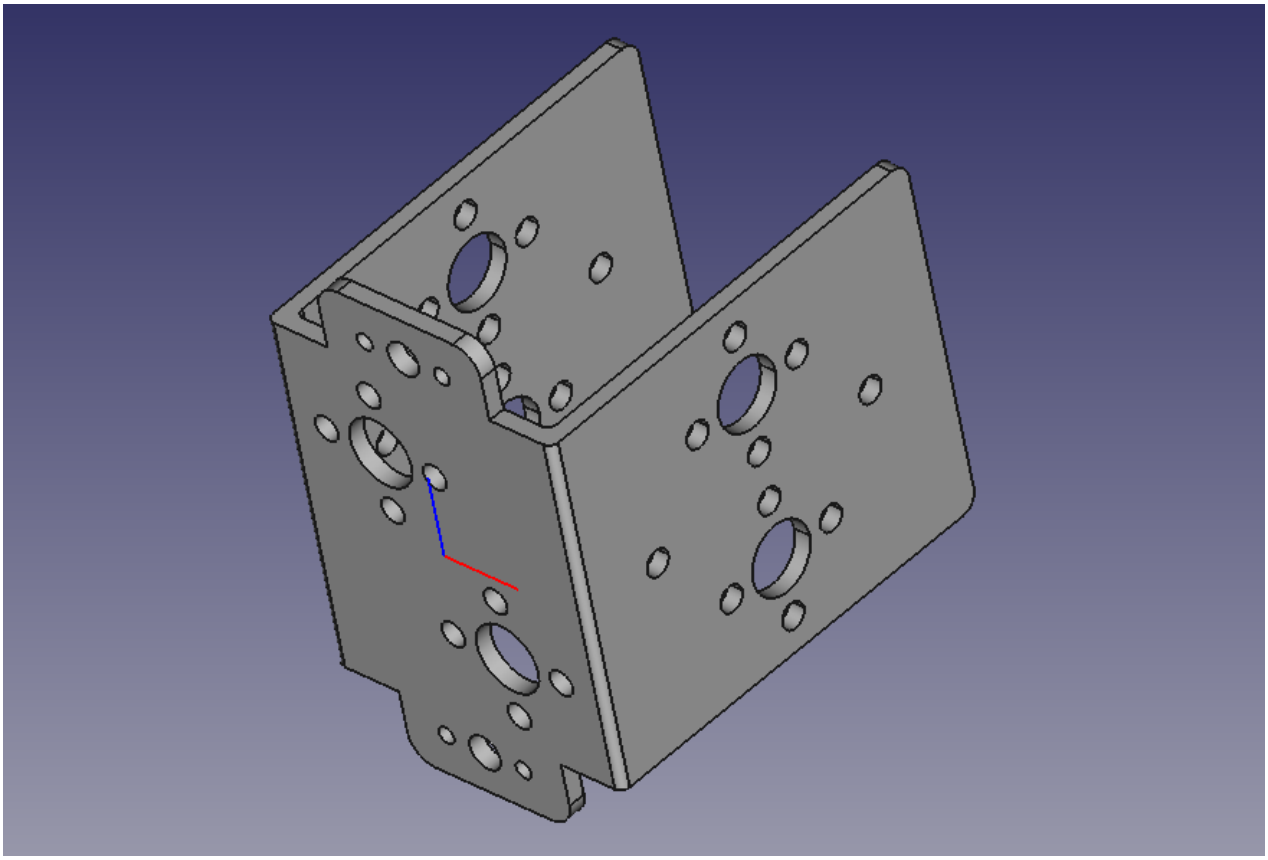

Рис.11 Кронштейн сервомотора

10. Включить в сборку Box фланец Flange. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a Part**) под именем Flange и в диалоге «**Place linked Part**» разместить Flange в сборке Box так, чтобы фланец оказался надетым на ротор сервомотора. Поэтому для Flange в качестве «**Parent Part**» выбрать LX-16A. Поворотами осей и смещениями по осям LCS_0 (Flange) относительно LCS_Rotor (LX-16A) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_0 (Flange) относительно LCS_Rotor (LX-16A) имеют следующие значения: X:0.0, Y:0.5, Z:1.0.

11. Включить в сборку Box кронштейн Holder. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Holder и в диалоге «**Place linked Part**» разместить Holder в сборке Box так, чтобы сервомотор с держателями оказался внутри кронштейна и ось ротора прошла через одно из отверстий кронштейна с обратной стороны сервомотора. Поэтому для Holder в качестве «**Parent Part**» выбрать LX-16A.

Поворотами осей и смещениями по осям LCS_0 (Holder) относительно LCS_0 (LX-16A) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_0 (Holder) относительно LCS_0 (LX-16A) имеют следующие значения: X:11.0, Y:9.5, Z:-18.5.

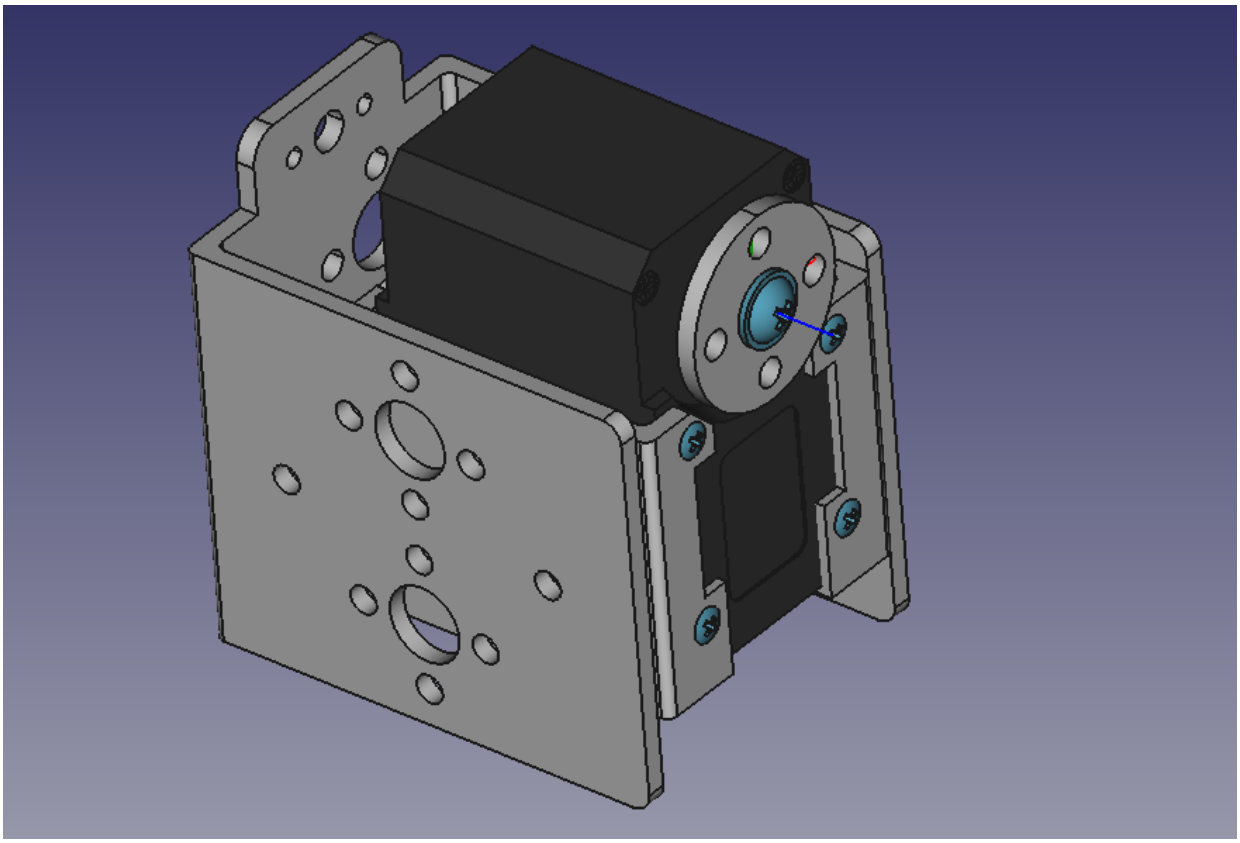

Рис.12. Узел сервомотора

12. Для визуального эффекта завершенности вставим 8 винтов M2x4 в держатели сервомотора и один винт M3x8 по центру фланца. Для этого при выбранном верстаке Assembly-4 с помощью инструмента «**Insert a Screw in the Assembly**» в диалоге «**Change fasteners parameters**» выбираем из списка тип винта «Fastener type» (например ISO7047), его диаметр (например M2) и длину винта (например 4mm). Модель винта появляется в центре LCS_Origin сборки. Выбрав винт в дереве сборки вызвать (**Right-Click**) инструмент **Transform** и интерактивно переместить изображение винта в нужное место. Выйти из Transform (**Esc**). Нужно следить, чтобы добавляемые винты находились в контейнере Model дерева сборки. Если винт оказывается вне контейнера Model, например при выполнении операции **Copy&Paste**, то перенести его вручную внутрь контейнера Model операцией **Drag&Drop**.

    Для того, чтобы винт M3x8 при анимации фланца вращался вместе с фланцем, необходимо привязать его к фланцу. Для этого выбрать винт M3x8 в дереве сборки и с помощью инструмента **Assembly => Edit Placement of a Fasterner** или одноименной иконки на панели инструментов FreeCAD в диалоге «**Attach a Fasterner in the assembly**» выбрать привязку LCS_0 (Flange) и установить ориентацию винта M3x8.

13. Отключить визуализацию всех LCS элементов сборки, кроме LCS_Origin (Box) для исключения загромождения картинки сборки Box. Сборка узла сервомотора (Box) представлена на Рис.12.

14. Сохранить документ Box (**Ctrl+S**) и закрыть его.

4.5. Анимация узла сервомотора (Box)

Анимация узла сервомотора заключается во вращении фланца сервомотора штатным инструментом «**Animate Assembly**» верстака Assembly-4. Для этого потребуется выполнить ряд предварительных шагов.

1. Создать переменную в сборке Box под именем Angle типа Float с помощью инструмента **Assembly => Add variable** или одноименной иконки на панели инструментов FreeCAD. Эта переменная расположена по пути Variables.Angle сборки Box.
2. Создать копию LCS_Rotor (LX-16A) в корне сборки Box. Для этого выбрать элемент Box#LX-16A.LCS_Robot и с помощью инструмента **Assembly => Import Datum object** или одноименной иконки на панели инструментов FreeCAD создать копию этого элемента под именем LCS_Rotor в корне сборки Box.
3. Выбрать созданный элемент Box#LCS_Rotor и отредактировать в его Placement поле Angle. В качестве значения поля Angle установить ссылку на переменную сборки Box Variables.Angle.
4. Перепривязать фланец Flange с LCS_Rotor (LX-16A) на LCS_Rotor (Box) с помощью инструмента **Assembly => Edit Placement of a Part** или одноименной иконки на панели инструментов FreeCAD.
5. Стартовать (Run) анимацию сборки Box с помощью инструмента «**Animate Assembly**» установив в диалоге циклическое (loop) изменение угла Angle с 0 до 360 градусов с шагом в один градус. В результате фланец и его винт крепления начнут вращаться вокруг оси ротора сервомотора.
6. Остановить анимацию, сохранить документ Box (**Ctrl+S**) и закрыть его.

5. Проектирование звеньев манипулятора ArmorX

Рассмотрим проектирование сборок второго уровня Base_link, A1_link, A2_link, A3_link и A4_link, которые представляют собой звенья манипулятора ArmorX. При этом основное внимание уделим специфическим моментам их проектирования.

5.1. Сборка звена Base_link

Сборка Base_link представляет собой основание (базу) манипулятора ArmorX и состоит из одного узла сервомотора (Box) и двух опорных кронштейнов (Bracket), удерживающих узел сервомотора (Box) так, что его ось Z Box#LCS_Rotor направлена вертикально вверх вдоль оси Z сборки Base_link#LCS_Origin.

Создание детали опорного кронштейна (Bracket) аналогично созданию держателя (Keeper_1). Единственное отличие состоит в использовании инструментов «**Linear Pattern**» и «**Polar Pattern**» (**Part Design => Apply a pattern**) для размножения крепежных отверстий по прямой и по радиальной линиям соответственно. Поэтому без подробных объяснений процесса его проектирования приведем лишь конечный результат на Рис.13.

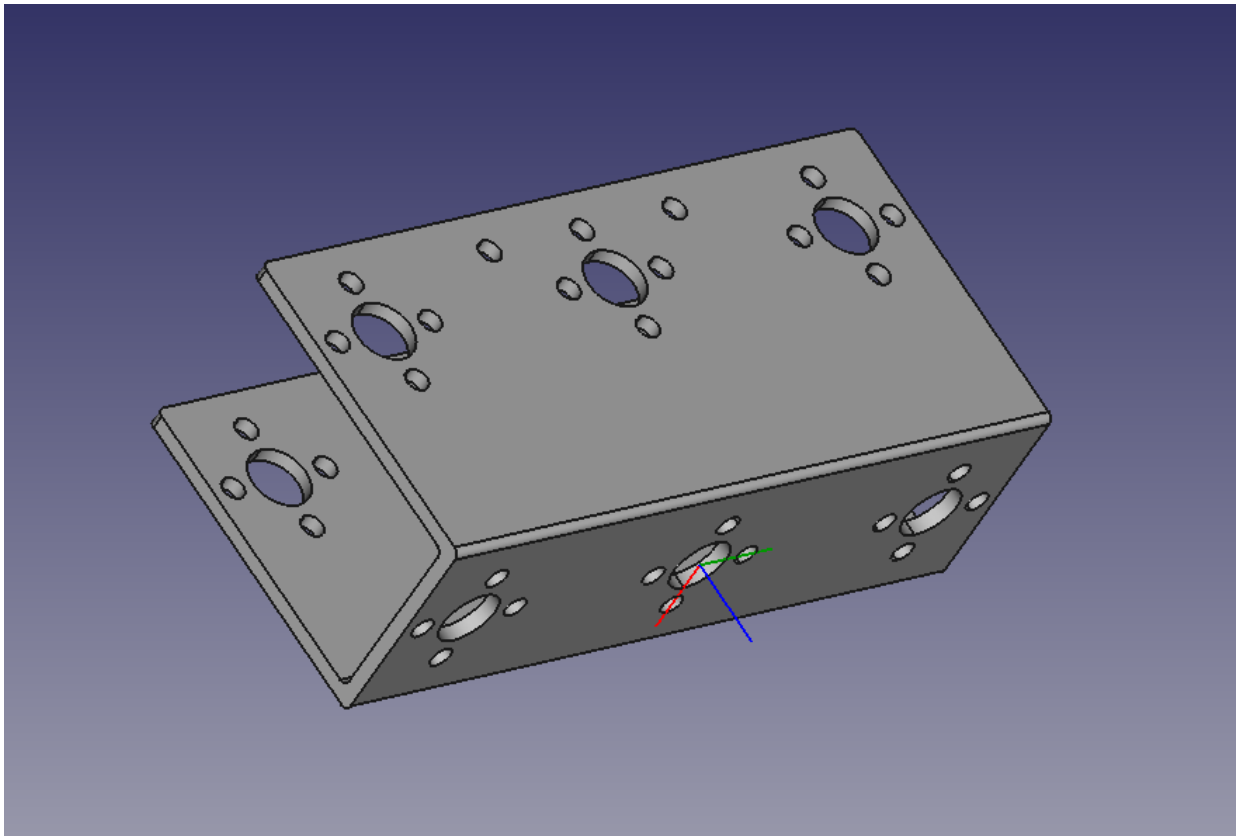

Рис.13. Опорный кронштейн

Проектирование сборки звена Base_link по шагам:

1. Создать новый документ FreeCAD (**Ctrl+N**)
2. Выбрать верстак **Assembly-4**
3. Создать в новом документе модель Model (**Ctrl+M** или **Assembly => New Model**).
4. Сохранить данный документ в файл (**Ctrl+S**) под именем Base_link. При этом в рабочем каталоге появится файл Base_link.FCStd.
5. Открыть (**Ctrl+O**) детали, из которых будет состоять сборка Base_link: узел сервомотора Box и опорный кронштейн Bracket.
6. Выбрать элемент Model в дереве документа и активировать его «**Toggle active part**» (**Right-Click**).

7. Включить в сборку Base_link узел сервомотора Box. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Box_A1 и в диалоге «**Place linked Part**» разместить Box_A1 в сборке Base_link так, что ось вращения ротора совпадет с осью Z LCS_Origin сборки Base_link. Для размещения Box_A1 выбрать в качестве «**Parent Part**» Parent Assembly и LCS_Origin (Box_A1) относительно LCS_Origin (Base_link). Сместить Box_A1 вверх по оси Z сборки Base_link на 60.0. Подтвердить размещение «**Ok**».

8. Включить в сборку Base_link опорный кронштейн Bracket. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Bracket_1 и в диалоге «**Place linked Part**» разместить Bracket_1 в сборке Base_link так, чтобы опорный кронштейн оказался сбоку от узла сервомотора Box_A1. Поэтому для Bracket_1 в качестве «**Parent Part**» выбрать Box_A1. Поворотами осей и смещениями по осям LCS_0 (Bracket_1) относительно LCS_Origin (Box_A1) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_0 (Bracket_1) относительно LCS_Origin (Box_A1) имеют следующие значения: X:18.5, Y:-0.0, Z:-40.0.

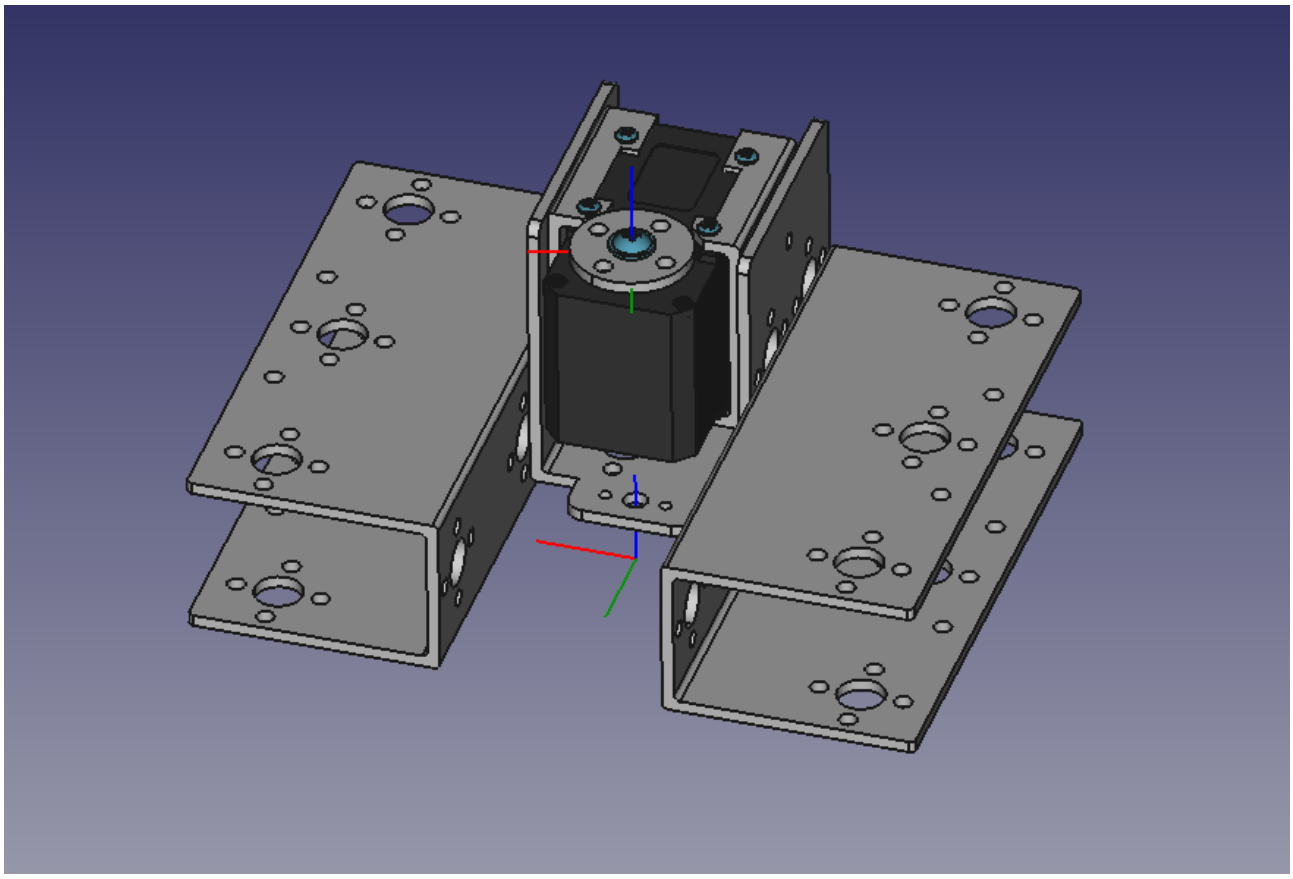

Рис.14. Звено Base_link

9. Включить в сборку Base_link второй опорный кронштейн Bracket по аналогии с первым опорным кронштейном Bracket_1, но с противоположной стороны узла сервомотора Box_A1. В качестве подсказки, смещения LCS_0 (Bracket_2)

относительно LCS_Origin (Box_A1) имеют следующие значения: X:-18.5, Y:0.0, Z:-40.0.

10. Отключить визуализацию всех LCS элементов сборки Base_link, кроме LCS_Origin (Base_link) и LCS_Rotor (Box_A1) для исключения загромождения картинки сборки Base_link. Сборка узла сервомотора (Base_link) представлена на Рис.14.

11. Сохранить документ Base_link (**Ctrl+S**) и закрыть его.

5.2. Сборка звена A1_link

Сборка A1_link представляет собой первое звено манипулятора ArmorX и состоит только из узла сервомотора Box.

Проектирование сборки звена A1_link по шагам:

1. Создать новый документ FreeCAD (**Ctrl+N**)
2. Выбрать верстак **Assembly-4**
3. Создать в новом документе модель Model (**Ctrl+M** или **Assembly => New Model**).
4. Сохранить данный документ в файл (**Ctrl+S**) под именем A1_link. При этом в рабочем каталоге появится файл A1_link.FCStd.
5. Открыть (**Ctrl+O**) узел сервомотора Box.

6. Включить в сборку A1_link узел сервомотора Box. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Box_A1 и в диалоге «**Place linked Part**» выбрать для Box_A1 в качестве «**Parent Part**» Parent Assembly и LCS_Origin (Box_A1) относительно LCS_Origin (A1_link). Подтвердить размещение «**Ok**». Результат полностью совпадает с узлом сервомотора Box и представлен на Рис.12.

7. Сохранить документ A1_link (**Ctrl+S**) и закрыть его.

5.3. Сборка звена A2_link

Сборка A2_link представляет собой второе звено манипулятора ArmorX и состоит из двух кронштейнов звена Linker.

Создание детали кронштейна звена Linker аналогично созданию держателя Keeper_1. При этом создаются две дополнительные локальные системы координат LCS_1 и LCS_2 по центру боковых отверстий. Без подробных объяснений процесса проектирования приведем лишь конечный результат на Рис.15.

Проектирование сборки звена A2_link по шагам:

8. Создать новый документ FreeCAD (**Ctrl+N**)
9. Выбрать верстак **Assembly-4**
10. Создать в новом документе модель Model (**Ctrl+M** или **Assembly => New Model**).
11. Сохранить данный документ в файл (**Ctrl+S**) под именем A2_link. При этом в рабочем каталоге появится файл A2_link.FCStd.
12. Открыть (**Ctrl+O**) деталь кронштейна звена Linker.

13. Выбрать элемент Model в дереве документа и активировать его «**Toggle active part**» (**Right-Click**).

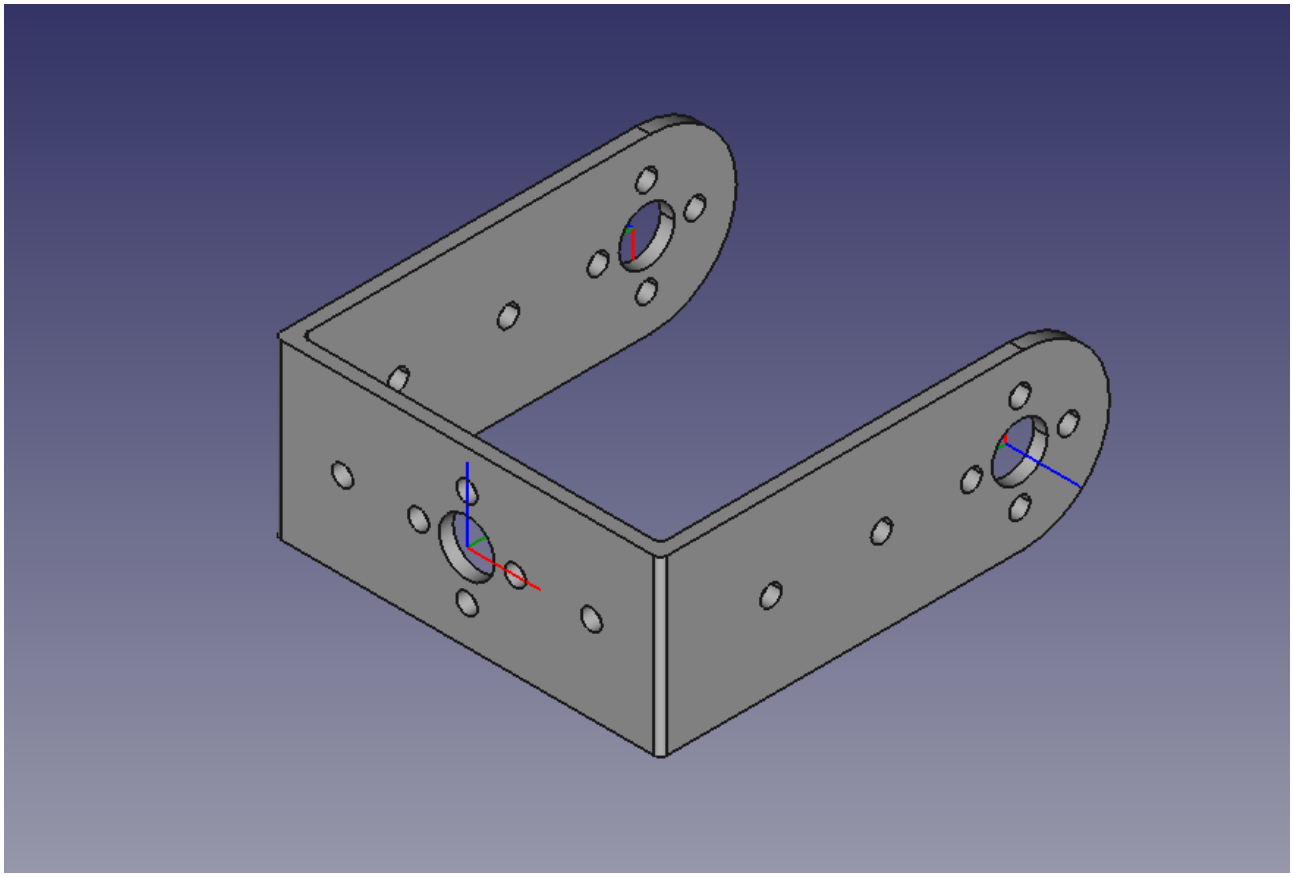

Рис.15. Кронштейн звена

14. Включить в сборку A2_link первый кронштейн Linker. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Linker_1 и в диалоге «**Place linked Part**» выбрать в качестве «**Parent Part**» Parent Assembly и LCS_0 (Linker_1) относительно LCS_Origin (A2_link). Подтвердить размещение «**Ok**».

15. Включить в сборку A2_link второй кронштейн Linker. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Linker_2 и в диалоге «**Place linked Part**» выбрать в качестве «**Parent Part**» Linker_1 и LCS_0 (Linker_2) относительно LCS_0 (Linker_1). Развернуть относительно оси Z Linker_2 на 180 градусов так, чтобы центральные отверстия кронштейнов совпали, а сами кронштейны образовали звено. Подтвердить размещение «**Ok**».

16. Для визуального эффекта завершенности вставим два винта M3x8 (ISO7045) и две гайки M3 (ISO4032) к ним в кронштейны звена Linker_1 и Linker_2 аналогично уже описанному в сборке узла сервомотора Box. Выполним привязку винтов к Linker_2 и привяжем гайки к Linker_1. Хотя это и не принципиально. Сборка звена A2_link представлена на Рис.16.

17. Сохранить документ A2_link (**Ctrl+S**) и закрыть его.

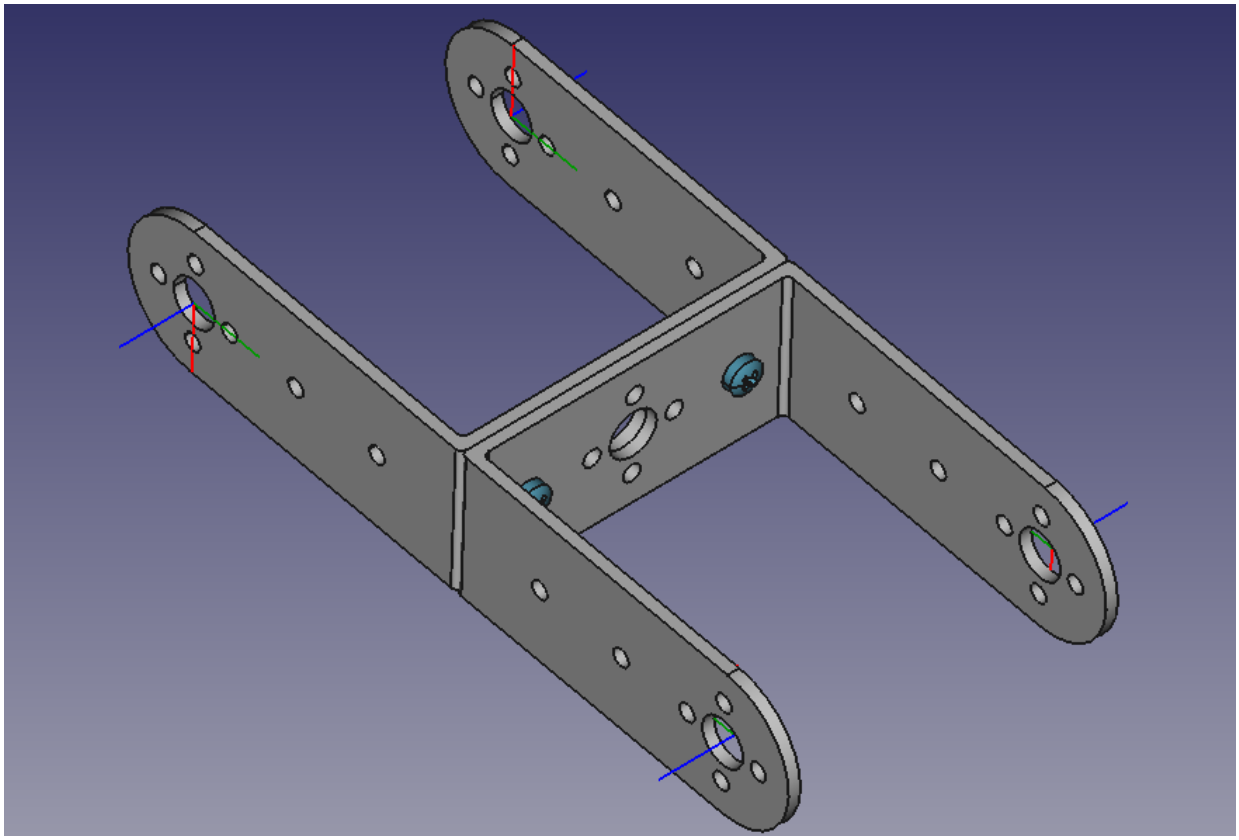

Рис.16. Звено A2_link

5.4. Сборка звена A3_link

Сборка A3_link представляет собой третье звено манипулятора ArmorX и состоит из двух узлов сервомотора Box и соединяющго их стержня CrossBar с двумя фланцами CrossFlange и двумя уголками Corner.

Создание стержня CrossBar и фланца CrossFlange полностью аналогично созданию фланца сервомотора Flange. Для стержня CrossBar при этом создаются две дополнительные локальные системы координат LCS_1 и LCS_2 по краям для крепления к стержню фланцев CrossFlange. Можно конечно было обойтись и без них, привязываясь исключительно к LCS_0 (в центре стержня), т. к. соединение фиксированное. Без подробных объяснений процесса проектирования приведем лишь конечный результат на Рис.17 и Рис.18.

Создание уголка Corner аналогично созданию держателя Keeper_1. При этом создаются две дополнительные локальные системы координат LCS_1 и LCS_2 для крепления к уголку узла сервомотора Box с одной стороны и фланца CrossFlange с другой стороны. Конечный результат приведен на Рис.19.

Проектирование сборки звена A3_link по шагам:

1. Создать новый документ FreeCAD (**Ctrl+N**)
2. Выбрать верстак **Assembly-4**
3. Создать в новом документе модель Model (**Ctrl+M** или **Assembly => New Model**).

4. Сохранить данный документ в файл (**Ctrl+S**) под именем A3_link. При этом в рабочем каталоге появится файл A3_link.FCStd.
5. Открыть (**Ctrl+O**) узлы и детали, из которых будет состоять сборка A3_link: узел сервомотора Box, стержень CrossBar, фланец стержня CrossFlange и уголок Corner.
6. Выбрать элемент Model в дереве документа и активировать его «**Toggle active part**» (**Right-Click**).

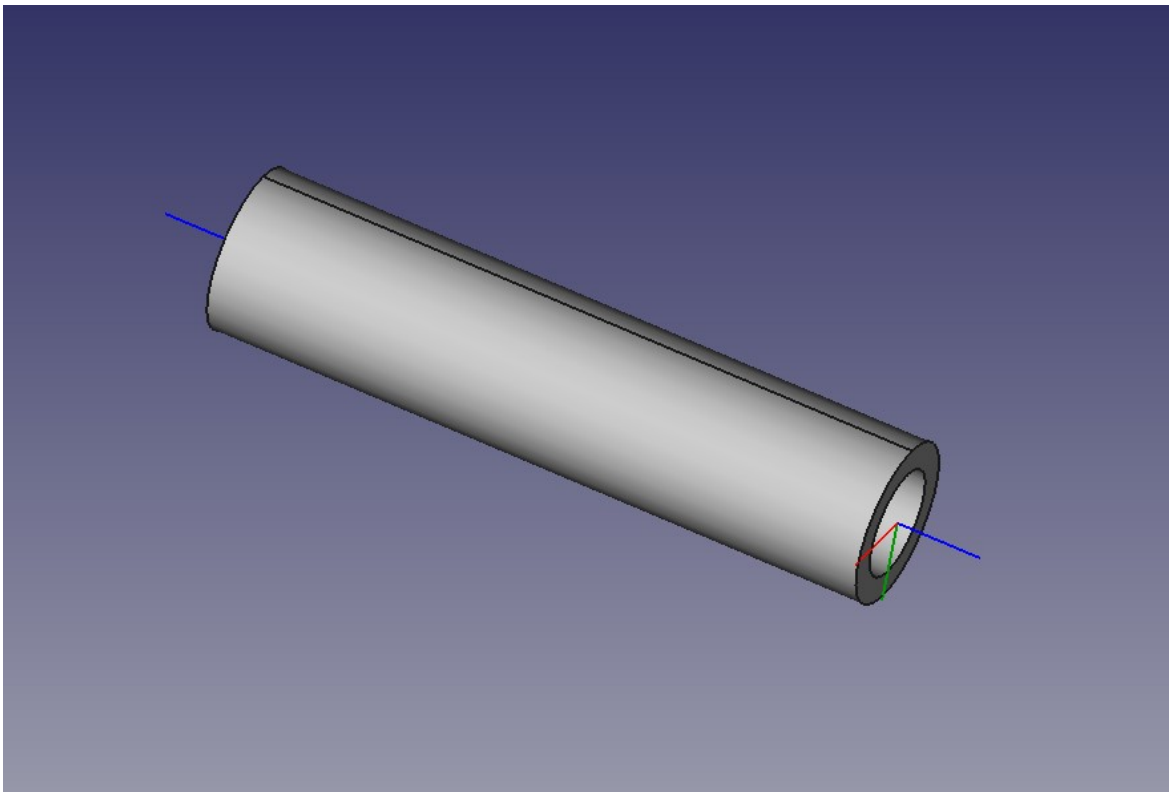

Рис.17. Стержень

7. Включить в сборку A3_link стержень CrossBar. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем CrossBar и в диалоге «**Place linked Part**» выбрать для CrossBar в качестве «**Parent Part**» Parent Assembly и LCS_0 (CrossBar) относительно LCS_Origin (A3_link). Подтвердить размещение «**Ok**». В центре сборки появится стержень с ориентацией вдоль оси Z LCS_Origin (A3_link). Вокруг этого стержня и будем наращивать сборку A3_link.

8. Включить в сборку A3_link первый фланец стержня CrossFlange. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем CrossFlange_1 и в диалоге «**Place linked Part**» выбрать для CrossFange_1 в качестве «**Parent Part**» CrossBar и LCS_0 (CrossFange) относительно LCS_1 (CrossBar). Фланец разместиться на одном конце стержня. Подкорректируем его размещение по оси Z на 4mm. Подтвердить размещение «**Ok**».

9. Включить в сборку A3_link второй фланец стержня CrossFlange. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем CrossFlange_2 и в

диалоге «**Place linked Part**» выбрать для CrossFange_2 в качестве «**Parent Part**» CrossBar и LCS_0 (CrossFange) относительно LCS_2 (CrossBar). Фланец разместиться на другом конце стержня. Подкорректируем его размещение по оси Z на 4mm. Подтвердить размещение «**Ok**».

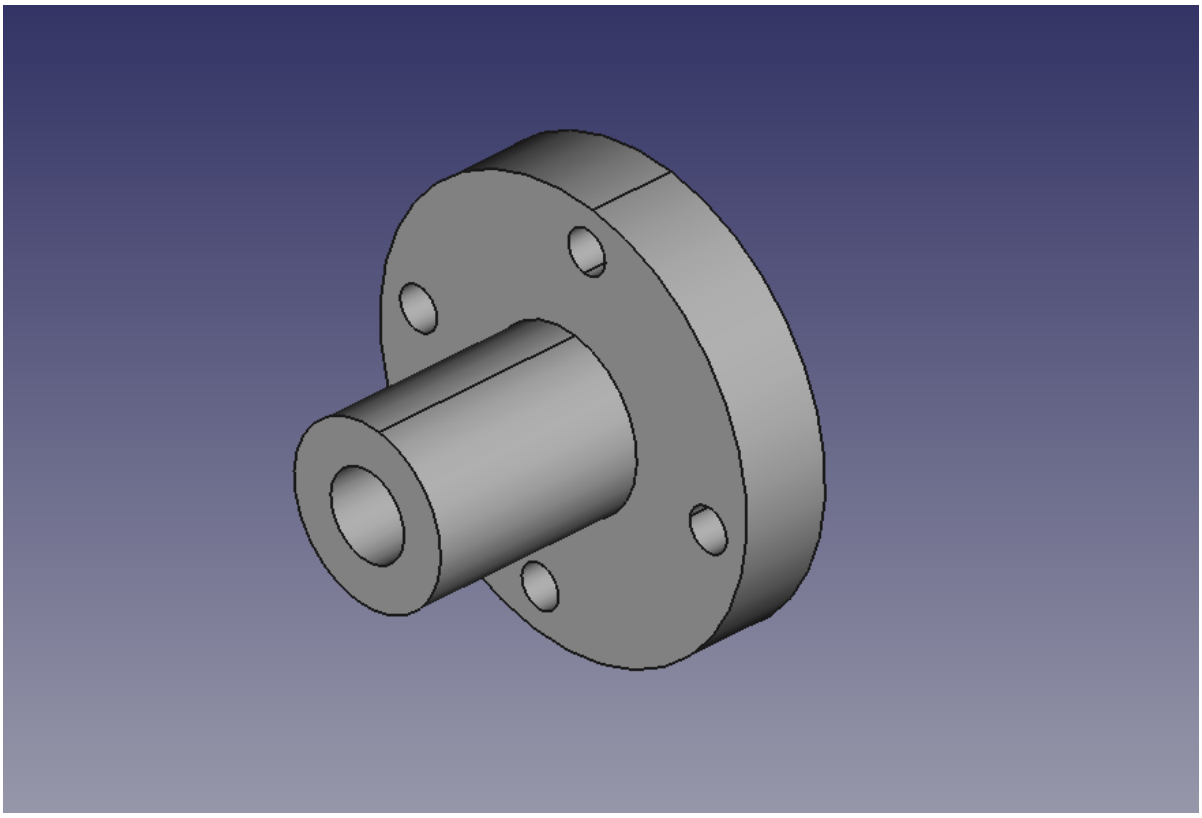

Рис.18. Фланец стержня

10. Включить в сборку A3_link первый уголок Corner. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Corner_1 и в диалоге «**Place linked Part**» выбрать для Corner_1 в качестве «**Parent Part**» CrossFlange_1 и LCS_2 (Corner_1) относительно LCS_0 (CrossFlange_1). С помощью вращений вокруг оси X добиться того, чтобы уголок разместился на фланце стержня. Перемещения не требуются. Подтвердить размещение «**Ok**».

11. Включить в сборку A3_link второй уголок Corner. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Corner_2 и в диалоге «**Place linked Part**» выбрать для Corner_2 в качестве «**Parent Part**» CrossFlange_2 и LCS_2 (Corner_2) относительно LCS_0 (CrossFlange_2). С помощью вращений вокруг оси X добиться того, чтобы уголок разместился на фланце стержня. Перемещения не требуются. Подтвердить размещение «**Ok**».

12. Включить в сборку A3_link первый узел сервомотора Box. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Box_A3 и в диалоге «**Place linked Part**» разместить Box_A3 в сборке A3_link так, чтобы монтажные отверстия уголка совпали с монтажными отверстиями кронштейна Holder узла сервомотора

Box_A3. Для размещения Box_A3 выбрать в качестве «**Parent Part**» CrossFlange_1 и LCS_Origin (Box_A3) относительно LCS_1 (CrossFlange_1). Поворотами осей и смещениями по осям LCS_Origin (Box_A3) относительно LCS_1 (CrossFlange_1) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_Origin (Box_A3) относительно LCS_1 (CrossFlange_1) имеют следующие значения: X:-6.0, Y:34.0, Z:-53.0.

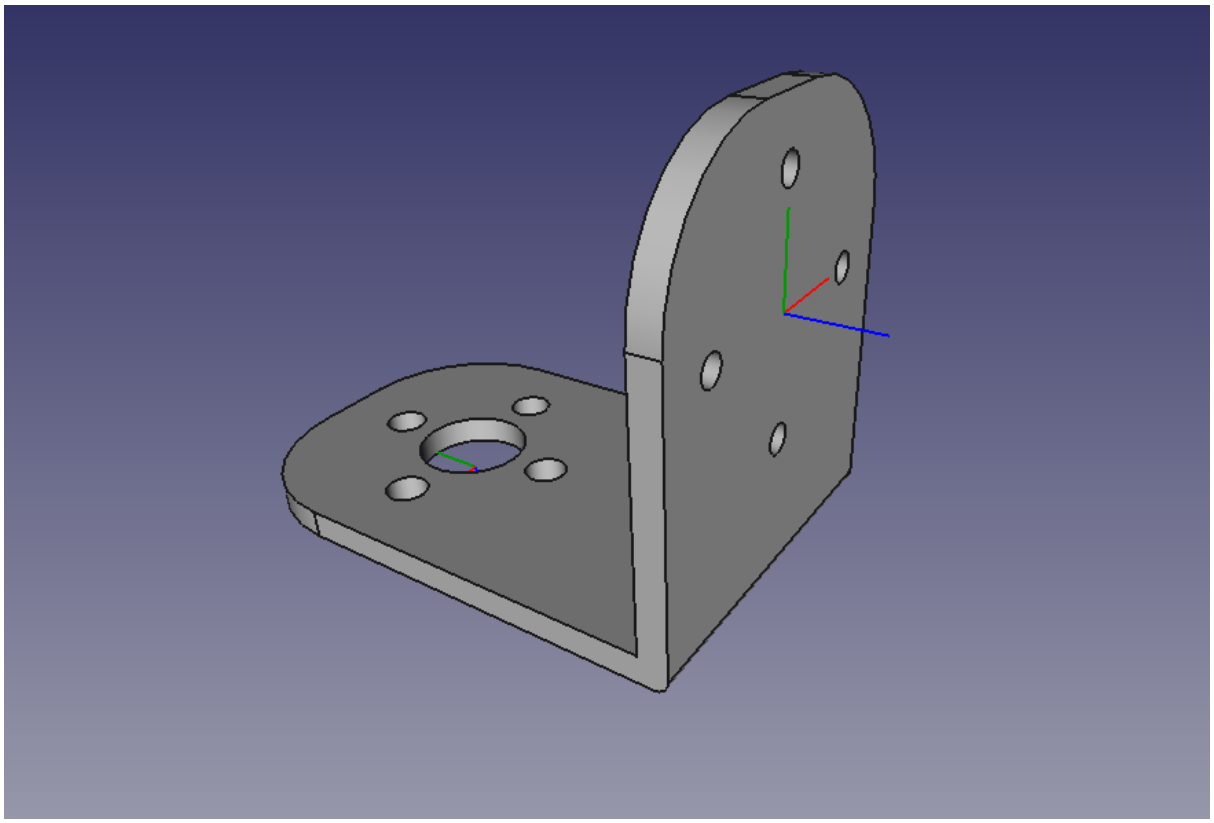

Рис.19. Уголок

13. Аналогично включить в сборку A3_link второй узел сервомотора Box под именем Box_A4 с подключением к CrossFlange_2.

14. Для визуального эффекта завершенности вставим четыре винта M2x8 (ISO7045) прикрепляющие уголок Corner_1 к фланцу стержня CrossFlange_1 и четыре винта прикрепляющие уголок Corner_2 к фланцу стержня CrossFlange_2. Выполним привязку винтов к Corner_1 и Corner_2 соответственно. При этом для удобства установки винтов узлы сервомоторов Box_A3 и Box_A4 можно временно сделать невидимыми (**Space**).

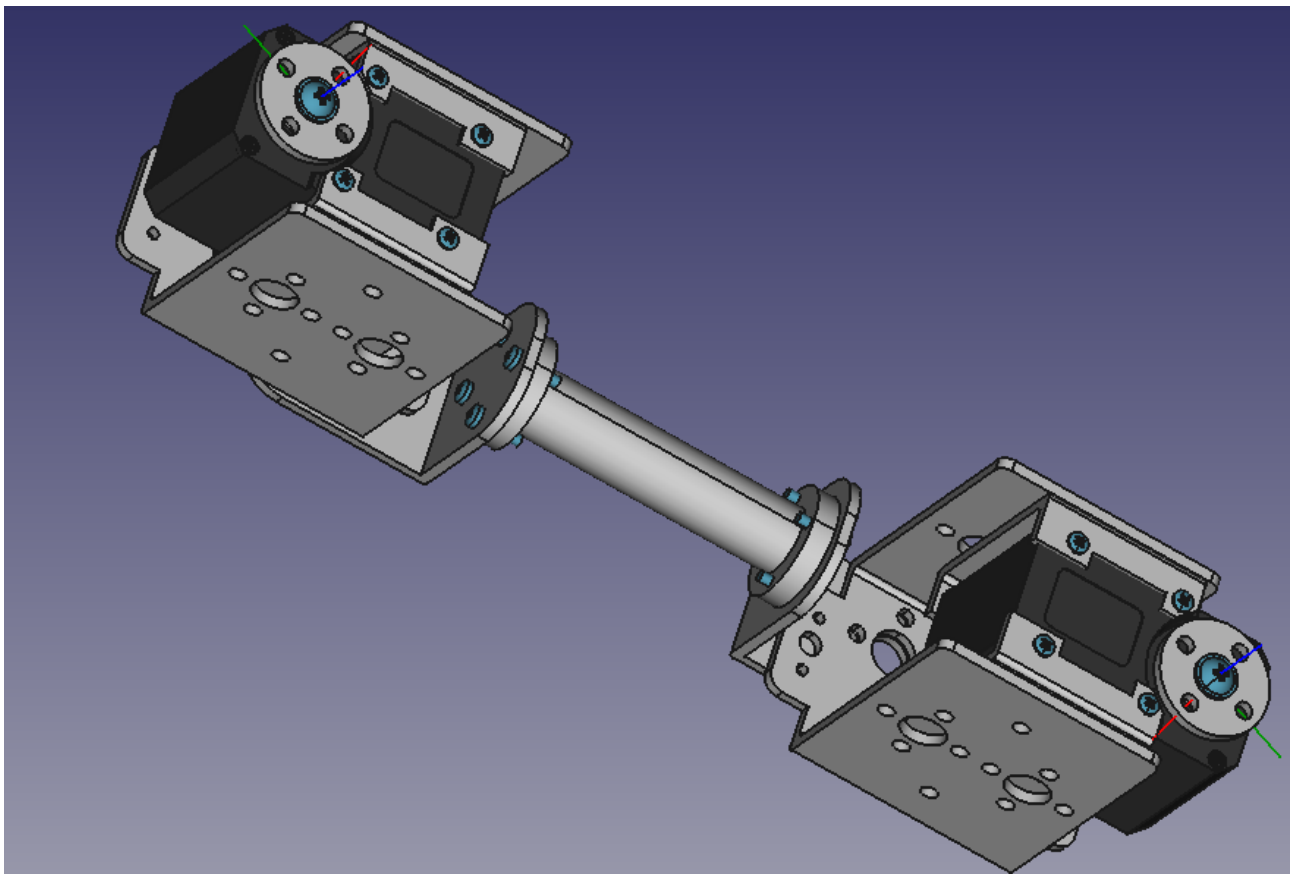

Рис.20. Звено A3_link

Для полноты картины приведем пошаговую инструкцию установки одного винта M2x8 на Corner_1:

1) Вызываем инструмент «**Fasteners**» (**Assembly => Fasterners** или одноименная иконка на панели инструментов FreeCAD).
2) В диалоге «**Change fastener parameters**» установить значения:
    1. Fasterner type: ISO7045
    2. Diameter: M2
    3. Length: 8
    4. «**Ok**»
3) Вызываем инструмент «**Edit Placement of a Fastener**» (**Assembly => Edit Placement of a Fastener** или одноименная иконка на панели инструментов FreeCAD).
4) В диалоге «**Attach a Fastener in the Assembly**» установить значения:
    1. Attach to: Corner_1
    2. Select attachment LCS in parent Part: LCS_2
    3. Z translation: -2
    4. Rotate X +90: два раза (180 градусов)
    5. X translation (Y translation): 8 или -8
    6. «**Ok**»

15. Отключить визуализацию всех LCS элементов сборки A3_link, кроме LCS_Rotor (Box_A3 и Box_A4) для исключения загромождения картинки сборки A3_link. Сборка звена A3_link представлена на Рис.20.

16. Сохранить документ A3_link (**Ctrl+S**) и закрыть его.

5.5. Сборка звена A4_link

Сборка A4_link представляет собой четвертое звено манипулятора ArmorX и состоит из узла сервомотора Box и одного кронштейна звена Linker.

Проектирование сборки звена A4_link по шагам:

1. Создать новый документ FreeCAD (**Ctrl+N**)
2. Выбрать верстак **Assembly-4**
3. Создать в новом документе модель Model (**Ctrl+M** или **Assembly => New Model**).
4. Сохранить данный документ в файл (**Ctrl+S**) под именем A4_link. При этом в рабочем каталоге появится файл A4_link.FCStd.
5. Открыть (**Ctrl+O**) кронштейн звена Linker и узел сервомотора Box.
6. Выбрать элемент Model в дереве сборки A4_link и активировать его «**Toggle active part**» (**Right-Click**).

7. Включить в сборку A4_link кронштейн Linker. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Linker и в диалоге «**Place linked Part**» выбрать в качестве «**Parent Part**» Parent Assembly и LCS_0 (Linker) относительно LCS_Origin (A4_link). Подтвердить размещение «**Ok**».

8. Включить в сборку A4_link узел сервомотора Box. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Box_A5 и в диалоге «**Place linked Part**» выбрать для Box_A5 в качестве «**Parent Part**» кронштейн звена Linker и LCS_Origin (Box_A5) относительно LCS_0 (Linker). Поворотами осей и смещениями по осям LCS_Origin (Box_A5) относительно LCS_0 (Linker) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_Origin (Box_A5) относительно LCS_0 (Linker) имеют следующие значения: X:-24.0, Y:-50.0, Z:0.0.
9. Отключить визуализацию всех LCS сборки A4_link, кроме LCS_Rotor (Box_A5), LCS_1 (Linker) и LCS_2 (Linker) для исключения загромождения картинки сборки A4_link. Сборка звена A4_link представлена на Рис.21.

10. Сохранить документ A4_link (**Ctrl+S**) и закрыть его.

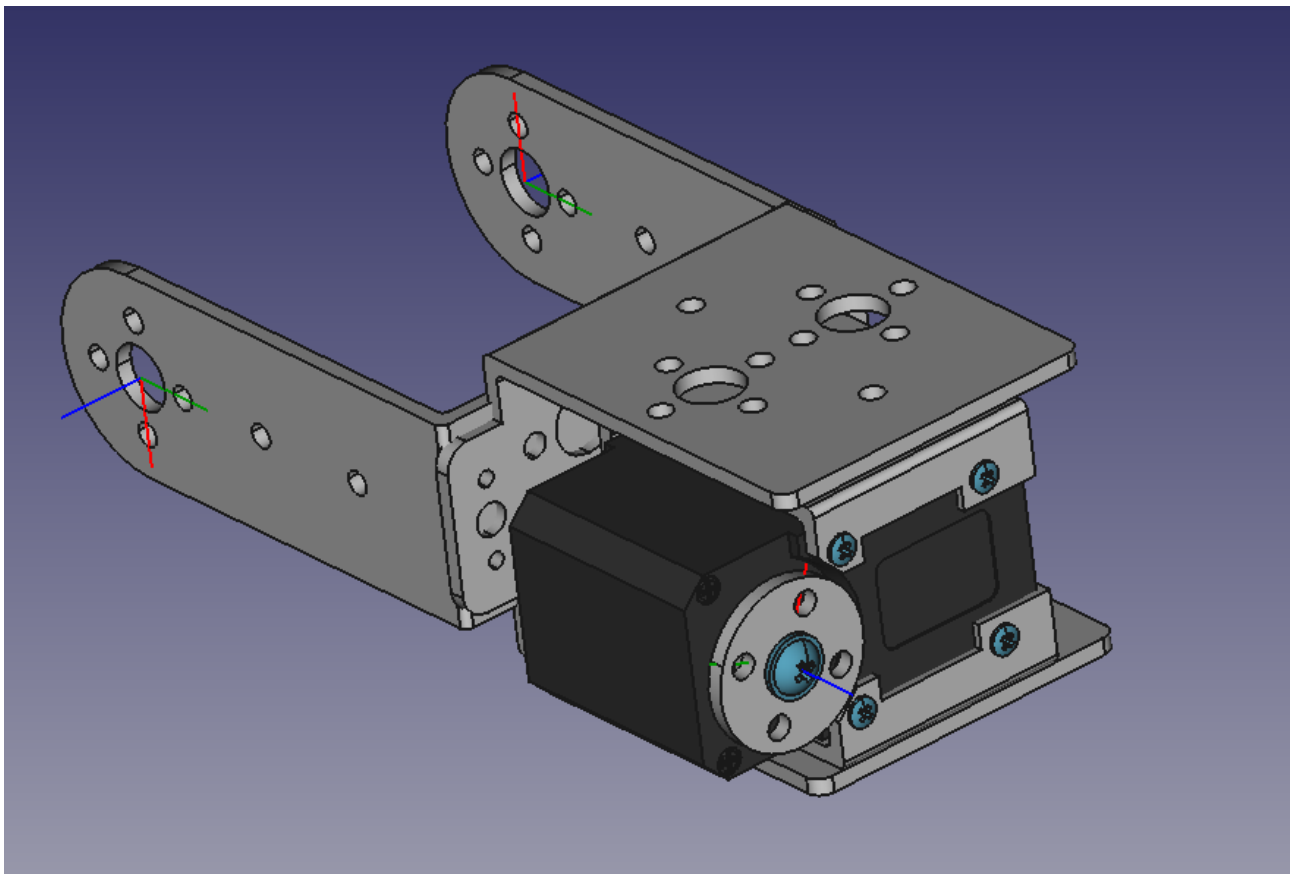

Рис.21. Звено A4_link

6. Проектирование манипулятора ArmorX

Манипулятор ArmorX состоит из звеньев Base_link, A1_link, A2_link, A3_link и A4_link, подключаемых друг к другу по цепочке снизу вверх так, что положение вышестоящего звена зависит от положений нижестоящих звеньев.

Assembly-4 FreeCAD предлагает два способа создания сборок, оба основанных на использовании LCS:

1. Первый способ предусматривает размещение в сборке (Model) деталей (Part) и/или ссылок (Link) совмещением их локальных систем координат (LCS) относительно друг друга.
2. Второй способ предусматривает использование «скелетного» эскиза (Master Sketch), в вершинах которого размещаются локальные системы координат (LCS) сборки, к которым привязываются LCS деталей (Part) и/или ссылок (Link).

Для сборки манипулятора ArmorX будем использовать первый способ. Для этого будем создавать дополнительные LCS звеньев с учетом следующих положений:

1. Удобства и логичности подключения звеньев друг к другу.

2. Реализации сочленений между звеньями, которые будут использованы в дальнейшем при анимации манипулятора ArmorX. Манипулятор ArmorX имеет тип 5-DOF и содержит пять сочленений A1, A2, A3, A4 и A5.

6.1. Сборка манипулятор ArmorX

Сборка манипулятора ArmorX по шагам:

1. Создать новый документ FreeCAD (**Ctrl+N**)
2. Выбрать верстак **Assembly-4**
3. Создать в новом документе модель Model (**Ctrl+M** или **Assembly => New Model**).
4. Сохранить данный документ в файл (**Ctrl+S**) под именем ArmorX. При этом в рабочем каталоге появится файл ArmorX.FCStd.
5. Открыть (**Ctrl+O**) звенья Base_link, A1_link, A2_link, A3_link, A4_link, из которых состоит манипулятор ArmorX.
6. Выбрать элемент Model в дереве сборки ArmorX и активировать его «**Toggle active part**» (**Right-Click**).

7. Включить в сборку ArmorX первое звено Base_link так, чтобы ось ротора сервомотора Box_A1 смотрела вертикально вверх вдоль оси Z (LCS_Origin) сборки ArmorX. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем Base_link и в диалоге «**Place linked Part**» выбрать в качестве «**Parent Part**» Parent Assembly и LCS_Origin (Base_link) относительно LCS_Origin (ArmorX). Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_Origin (Base_link) относительно LCS_Origin (ArmorX) имеют следующие значения: X:0.0, Y:0.0, Z:-6.0.

8. Второе звено A1_link крепится к фланцу сервомотора Box_A1 (LCS_Rotor) первого звена Base_link. Однако LCS_Rotor находится в сборке Box_A1 и не может быть непосредственно использован для привязки звена A1_link к звену Base_link. Поэтому требуется создать копию LCS_Rotor (Box_A1) в сборке Base_link под именем LCS_A1. Для этого выбрать LCS_Rotor в дереве документа Base_link и с помощью инструмента «**Import Datum object**» (**Assembly => Import Datum object** или одноименной иконки на панели инструментов FreeCAD) создать его копию под именем LCS_A1 в корне сборки Base_link. По сути LCS_A1 представляет собой первое сочленение A1 манипулятора ArmorX.

9. Включить в сборку ArmorX второе звено A1_link так, чтобы ось ротора сервомотора Box_A2 была перпендикулярна оси Z (LCS_A1) сборки Base_link. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем A1_link и в диалоге «**Place linked Part**» выбрать в качестве «**Parent Part**» Base_link и LCS_Origin (A1_link) относительно LCS_A1 (Base_link). Поворотами осей и смещениями по осям LCS_Origin (A1_link) относительно LCS_A1 (Base_link) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_Origin (A1_link) относительно LCS_A1 (Base_link) имеют следующие значения: X:25.5, Y:14.0, Z:19.5.

10. Третье звено A2_link крепится к фланцу сервомотора Box_A2 (LCS_Rotor) второго звена A1_link. Однако LCS_Rotor находится в сборке Box_A2 и не может быть непосредственно использован для привязки звена A2_link к звену A1_link. Поэтому требуется создать копию LCS_Rotor (Box_A2) в сборке Base_link под именем LCS_A2. Для этого выбрать LCS_Rotor в дереве документа A1_link и с помощью инструмента «**Import Datum object**» (**Assembly => Import Datum object** или одноименной иконки на панели инструментов FreeCAD) создать его копию под именем LCS_A2 в корне сборки A1_link. По сути LCS_A2 представляет собой второе сочленение A2 манипулятора ArmorX.

11. Для подключения звена A2_link к соседним звеньям A1_link и A3_link предварительно сделать четыре вспомогательных LCS в местах его подключения. Для этого скопировать в корень A2_link с помощью инструмента «**Import Datum object**» LCS_1 и LCS_2 деталей Linker_1 и Linker_2 под именами LCS_11, LCS_12, LCS_21 и LCS_22.

12. Включить в сборку ArmorX третье звено A2_link так, чтобы одно его центральное монтажное отверстие (LCS_11) совпало с осью ротора сервомотора Box_A2 (LCS_A2) сборки A1_link. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем A2_link и в диалоге «**Place linked Part**» выбрать в качестве «**Parent Part**» A1_link и LCS_11 (A2_link) относительно LCS_A2 (A1_link). Поворотами осей и смещениями по осям LCS_11 (A2_link) относительно LCS_A2 (A1_link) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_11 (A2_link) относительно LCS_A2 (A1_link) имеют следующие значения: X:0.0, Y:0.0, Z:1.0.

13. Для подключения звена A3_link к соседним звеньям A2_link и A4_link предварительно сделать два сочленения LCS_A3 и LCS_A4 в местах его подключения. Для этого скопировать в корень A3_link с помощью инструмента «**Import Datum object**» LCS_Rotor его вложенных сборок Box_A3 и Box_A4 под именами LCS_A3 и LCS_A4 соответственно. По сути LCS_A3 и LCS_A4 представляют собой третье A3 и четвертое A4 сочленения манипулятора ArmorX.

14. Включить в сборку ArmorX четвертое звено A3_link так, чтобы ось ротора сервомотора Box_A3 (LCS_A3) сборки A3_link совпала с центральным монтажным отверстием (LCS_22) сборки A2_link. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем A3_link и в диалоге «**Place linked Part**» выбрать в качестве «**Parent Part**» A2_link и LCS_A3 (A3_link) относительно LCS_22 (A2_link). Поворотами осей и смещениями по осям LCS_A3 (A3_link) относительно LCS_22 (A2_link) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_A3 (A3_link) относительно LCS_22 (A2_link) имеют следующие значения: X:0.0, Y:0.0, Z:-3.0.

15. Для подключения звена A4_link к звену A3_link предварительно сделать два вспомогательных LCS в местах его подключения. Для этого скопировать в корень A4_link с помощью инструмента «**Import Datum object**» LCS_1 и LCS_2 детали Linker под именами LCS_1 и LCS_2 соответственно.

16. Включить в сборку ArmorX пятое звено A4_link так, чтобы одно его центральное монтажное отверстие (LCS_1) совпало с осью ротора сервомотора Box_A4 (LCS_A4) сборки A3_link. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем A4_link и в диалоге «**Place linked Part**» выбрать в качестве «**Parent Part**» A3_link и LCS_1 (A4_link) относительно LCS_A4 (A3_link). Поворотами осей и смещениями по осям LCS_1 (A4_link) относительно LCS_A4 (A3_link) добиться результата. Подтвердить размещение «**Ok**». В качестве подсказки, смещения LCS_1 (A4_link) относительно LCS_A4 (A3_link) имеют следующие значения: X:0.5, Y:0.0, Z:0.0.

17. В завершение сборки ArmorX реализуем его сочленение A5. Именно к нему будет крепиться захват, который в данной статье не рассматривается. Для этого скопировать в корень A4_link с помощью инструмента «**Import Datum object**» LCS_Rotor его вложенной сборки Box_A5 под именем LCS_A5. По сути LCS_A5 представляет собой пятое A5 сочленение манипулятора ArmorX.

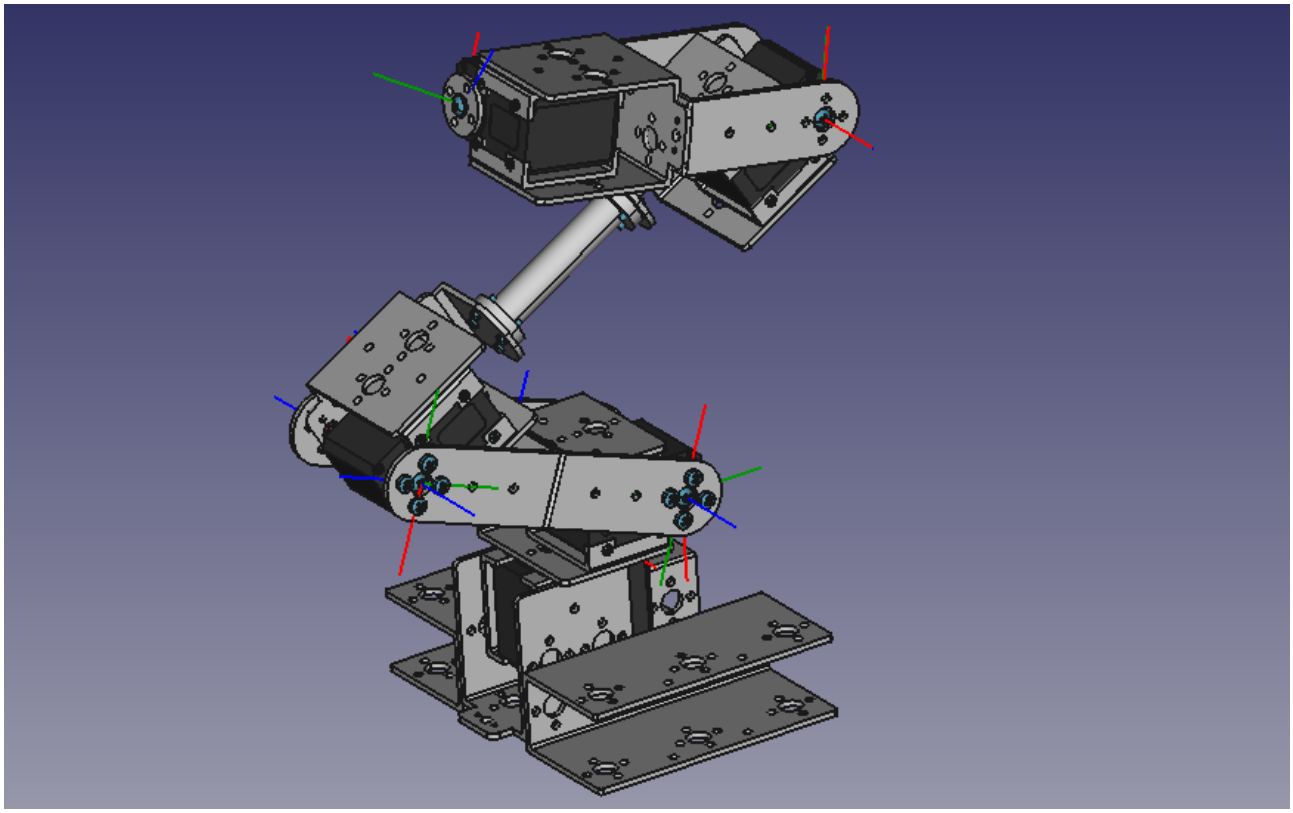

Рис.22. Манипулятор ArmorX

6.2. Дополнительные элементы сборки

Под дополнительными элементами сборки будем понимать прежде всего элементы крепежа: винты, гайки и шайбы. Assembly-4 предлагает встроенный интерфейс к верстаку Fasterners, с помощью которого в сборку вставляются элементы крепежа. Сборка Assembly-4 имеет контейнер Patrs на том, же уровне что и Model. Элементы крепежа, используемые в сборке, рекомендуется встраивать в компоненты App::Part, которые находятся в этом контейнере Parts.

Для наглядности рассмотрим установку винта M3x7 (ISO7045) с гайкой M3 (ISO4032) в монтажное отверстие уголка Corner_1 и узла сервомотора Box_A3 звена A3_link. Для начала необходимо создать детали M3x7_Screw и M3_Nut со встроенным в них винтом M3x7 и гайкой M3 соответственно. В дальнейшем выполнять установку именно этих деталей M3x7_Screw и M3_Nut.

1. Открыть манипулятор ArmorX (**Ctrl+O**).
2. Выбрать верстак **Assembly-4**
3. Выбрать документ ArmorX.
4. Создать деталь под именем M3x7_Screw с помощью инструмента «**New Part**» (**Assembly => New Part** или одноименная иконка на панели инструментов FreeCAD). При этом в контейнере Parts появится новая деталь M3x7_Screw.
5. Выбрать деталь M3x7_Screw в дереве документа и с помощью инструмента «**Fasterners**» (**Assembly => Fasterners** или одноименная иконка на панели инструментов FreeCAD) открыть диалог «**Change fasterner parameters**».
6. В диалоге выбрать тип винта (Fasterner Type) ISO7045, его диаметр (Diameter) M3 и длину 7mm. Подтвердить выбор «**Ok**». В контейнере детали M3x7_Screw появится уже параметризованный винт M3x7-Screw.
7. Аналогичную процедуру повторить для создания детали M3_Nut гайки M3 типа ISO4032.
8. Включить в сборку ArmorX винт M3x7_Screw. Для этого сделать на него ссылку (**Ctrl+L** или **Assembly => Link a part**) под именем по умолчанию и в диалоге «**Place linked Part**» выбрать в качестве «**Parent Part**» A3_link и LCS_0 (M3x7_Screw) относительно LCS_11 (A3_link). Поворотами осей и смещениями по осям LCS_0 (M3x7_Screw) относительно LCS_11 (A3_link) добиться, чтобы винт занял свое место в монтажном отверстии уголка Corner_1. Подтвердить размещение «**Ok**».
9. Аналогично поступить с гайкой M3_Nut, которая должна оказаться на винту со стороны узла сервомотора Box_A3 звена A3_link.
10. Отключить видимость деталей M3x7_Screw и M3_Nut в контейнере Parts, оставив видимость ссылок на них в дереве манипулятора ArmorX.

Привязку деталей крепежа можно выполнять к различным LCS. Например, гайку можно привязать к LCS винта, а не к LCS уголка. Единственное условие заключается в том, чтобы эти LCS корректно перемещались при анимации манипулятора.

6.3. Анимация манипулятора ArmorX

Манипулятор ArmorX имеет пять степеней свободы (5-DOF), которые реализуются вращениями пяти его сочленений A1, A2, A3, A4 и A5. Анимация манипулятора позволяет достичь не только визуального эффекта, но и убедиться в корректности сборки узлов и самого манипулятора. Если звенья или детали во время движения «разъезжаются», то скорее всего ошибка кроется в их привязке.

Верстак Assembly-4 имеет встроенный инструмент «**Animate Assembly**» (**Assembly => Animate Assembly** или одноименная иконка на панели инструментов FreeCAD), который позволяет в диалоге выбирать одну переменную сборки верхнего уровня. В нашем случае эта переменная должна находится в контейнере ArmorX#Variables. В диалоге устанавливаются такие параметры анимации как интервал и шаг изменения переменной, циклическое (Loop) или периодическое (Pendulum) ее изменение. Если к данной переменной привязать вращение LCS конкретного сочленения, то получим анимацию этого сочленения.

Однако имеется три серьезных ограничения текущей реализации инструмента «**Animate Assembly**»:

1. Можно управлять только одной переменной.
2. Эта переменная должна находится в сборке верхнего уровня. Нельзя получить доступ к переменным, расположенным во вложенных сборках (внутренние переменные).
3. Если использовать эту переменную в вычисляемом выражении внутренней переменной, то это выражение имеет смысл только со сборкой верхнего уровня. Независимая компиляция вложенной сборки приведет к ошибке.

Инструмент «**Animate Assembly**», как собственно и весь верстак Assembly-4, реализован на Python и, следовательно, может быть при желании модифицирован. Однако, понимая его ограничения и чисто вспомогательный характер, для целей анимации манипулятора ArmorX пойдем несколько иным путем.

Реализуем в среде FreeCAD собственный скрипт на Python, который будет выполнять анимацию манипулятора ArmorX вращением всех его сочленений A1, A2, A3, A4 и A5 по определенной фиксированной логике. Данный скрипт оформим в виде Macro FreeCAD. Текст скрипта приведен в «**Приложение B**».

Прежде всего создадим пять переменных типа Float по количеству сочленений в соответствующих сборках звеньев: Base_link#Variables.A1_joint, A1_link#Variables.A2_joint, A3_link#Variables.A3_joint, A3_link#Variables.A4_joint, A4_link#Variables.A5_joint. Выполним привязку к ним угла поворота Angle размещения Placement LCS сочленений через ExpressionEngine:

1. Base_link#LCS_A1 => Angle = Variables.A1_joint
2. A1_link#LCS_A2 => Angle = Variables.A2_joint
3. A3_link#LCS_A3 => Angle = Variables.A3_joint
4. A3_link#LCS_A4 => Angle = Variables.A4_joint
5. A4_link#LCS_A5 => Angle = Variables.A5_joint

Файл скрипта под именем **armorx.py** скопировать в каталог ~/.FreeCAD/Macro. Вызов скрипта выполняется через инструмент FreeCAD «**Execute Macro**» (**Macro => Macros... => Execute Macro** или одноименную иконку на панели инструментов FreeCAD). В появившемся диалоге выбрать файл скрипта **armorx.py** и запустить его выполнение «**Execute**». По умолчанию, параметры анимации 0.2 секунды (delay) между кадрами в течение 100 секунд (duration). Если документ манипулятора ArmorX предварительно не открыт (**Ctrl+O**), скрипт **armrox.py** откроет его самостоятельно.

Логика скрипта анимации **armrox.py** достаточно очевидна и не требует подробного комментария. Заметим только, что скрипты Python выполняются в главном потоке FreeCAD и, следовательно, прямой вызов в этом потоке time.sleep вызовет «засыпание» самого FreeCAD, а не только скрипта. Поэтому цикл анимации выполняется в параллельном потоке, специально создаваемом для этого средством Thread. Суть скрипта заключается в изменении значений переменных A1_joint, A2_joint, A3_joint, A4_joint, A5_joint с определенным шагом A1_STEP, A2_STEP, A3_STEP, A4_STEP, A5_STEP в интервалах A1_LIMIT, A2_LIMIT, A3_LIMIT, A4_LIMIT , A5_LIMIT сответственно каждые delay секунд. Кадр анимации формируется вызовом функции recompute().

7. Заключение

Данная статья демонстрирует возможности параметрической САПР с открытым исходным кодом FreeCAD на примере полного цикла создания виртуальной 3D модели реального робота манипулятора ArmorX средствами верстака Assembly-4.

Особый интерес представляет использования FreeCAD при создании URDF-описания (Universal Robot Description Format) робота-манипулятора для разработки его системы управления (СУ) в среде ROS (Robot Operation System), включая визуализацию управления в Rviz и создание виртуального аппаратного симулятора манипулятора в Gazebo. Для этого, в идеале, требуется реализовать плагин FreeCAD на Python, который в автоматическом режиме будет генерировать URDF-описание манипулятора, исходя из его виртуальной 3D модели, включая его динамические параметры.

Данный вопрос, а также вопросы подготовки 3D-печати деталей манипулятора и использование в сборках «скелетных» эскизов (Master Sketch) верстака Assembly-4 FreeCAD пока остались за рамками этой статьи.

Приложения

A. Выражения (Expression) FreeCAD

Синтаксический стиль выражений:

a) Имена функций и имена переменных начинаются со строчной буквы.
b) Функции и переменные, имена которых начинаются с символа подчеркивания, являются внутренними и не должны использоваться.

c) Имена объектов начинаются с заглавной буквы.
d) Имена констант имеют только заглавные буквы.

1. Встроенные типы данных
   1) Целое число: 1, 2, -5, ...
   2) Число с плавающей точкой: 3.56, -23.54, ...
   3) Строка: "mystring", …
   4) Список: [12, "mystring", ...]
   5) Объект: MyDoc#Variables.length, MySketch.Constraints, ...
2. Математические константы
   1) **e** = 2.71828
   2) **pi** = 3.14159
3. Арифметические операции:
   1) Сложение: **+**
   2) Вычитание: **-**
   3) Умножение: ***
   4) Деление: **/**
   5) Остаток от деления: **%**
   6) Возведение в степень: **^**
4. Математические функции
   1) **sin(x)**, **cos(x)**, **tan(x)**, **asin(x)**, **acos(x)**, **atan(x)**, **sinh(x)**, **cosh(x)**, **tanh(x)**
   2) **exp(x)**, **log(x)**, **log10(x)**, **pow(x, y)**, **sqrt(x)**
   3) **abs(x)**, **ceil(x)**, **floor(x)**, **mod(x, y)**, **round(x)**, **trunc(x)**
5. Агрегатные функции
   Применяются к ячейкам (cell) глобальной таблицы переменных (spreadsheet).
   1) Среднее арифметическое от ячейки x до ячейки y: **average(x:y)**
   2) Количество ячеек между ячейкой x и ячейкой y: **count(x:y)**
   3) Максимальное значение ячеек между ячейкой x и ячейкой y: **max(x:y)**
   4) Минимальное значение ячеек между ячейкой x и ячейкой y: **min(x:y)**
   5) Дисперсия значений ячеек: **stddev(x:y)**
   6) Сумма значение ячеек: **sum(x:y)**
6. Арифметическое выражение
   Имеет стандартный синтаксис и общепринятый порядок приоритетов, включая круглые скобки.
7. Логические операции
   1) Равно: **==**
   2) Не равно: **!=**
   3) Больше: **>**
   4) Меньше: **<**
   5) Больше или равно: **>=**
   6) Меньше или равно: **<=**
8. Логическое выражение

Имеет стандартный синтаксис и общепринятый порядок приоритетов, включая круглые скобки.

9. Условное выражение: **Condition ? ResultTrue : ResultFalse**
10. Единицы измерения
    1) Угол
        a) Градус: **deg**
        b) Радиан: **rad**
        c) Градиан: **gon**
    2) Длина
        a) Нанометр: **nm**
        b) Микрон: **um**
        c) Миллиметр: **mm**
        d) Сантиметр: **cm**
        e) Дециметр: **dm**
        f) Метр: **m**
        g) Километр: **km**
        h) Дюйм: **in**
        i) Фут: **ft**
        j) Ярд: **yd**
    3) Сила
        a) Милиньютон: **mN**
        b) Ньютон: **N**
        c) Килоньютон: **kN**
    4) Вес
        a) Микрограмм: **ug**
        b) Миллиграмм: **mg**
        c) Грамм: **g**
        d) Килограмм: **kg**
        e) Тонна: **t**
        f) Фунт: **lb**
    5) Время
        a) Секунда: **s**
        b) Минута: **min**
        c) Час: **h**
    6) Частота
        a) Герц: **Hz**
        b) Килогерц: **kHz**
        c) Мегагерц: **MHz**
    7) Крутящий момент
        a) Ньютон метр: **Nm**
    8) Объем
        a) Миллилитр: **ml**
        b) Литр: **l**
    9) Сила тока

a) Миллиампер: **mA**
b) Ампер: **A**
c) Килоампер: **kA**

11. Объекты документа

    Документ FreeCAD представляет собой структуру дерева, узлами которого являются объекты деталей, ссылок, сборок, эскизов, таблиц, операций и прочее.

    Доступ к полям данных объектов документа возможен в общем виде:
    Document#Element_1.Element_2. … Element_n.Data
    где:
    Document — имя документа (имя файла документа)
    Element_i — i-й элемент ветки дерева документа от его корня, i = 1,…,n
    Data — поле данных последнего элемента Element_n

12. Глобальная таблица переменных
    FreeCAD на данный момент не имеет концепции глобальных переменных. Вместо этого переменные могут быть определены как ячейки в глобальной таблице с использованием верстака Spreadsheet с заданием их имени в качестве алиаса (alias). Доступ к таким переменным возможен из любого выражения по имени в стиле доступа к полям объекта.

B. Скрипт анимации манипулятора ArmorX

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#*******************************************
# Анимация манипулятора ArmorX (5-DOF) в среде FreeCAD 0.19
# 12.2020, Скребцов В.И. v.skrebtsov@mail.ru
#
#*******************************************

import time
from threading import Thread

import FreeCAD as App
from FreeCAD import Gui

# Шаг изменения угла сочленения в градусах
A1_STEP = 1.0
A2_STEP = 2.0
A3_STEP = 2.0
A4_STEP = 3.0
A5_STEP = 1.0
# Границы изменения угла сочленения в градусах
A1_LIMIT = [-360, 360]
A2_LIMIT = [90, 270]
A3_LIMIT = [-120, 120]
```

```
A4_LIMIT = [-120, 120]
A5_LIMIT = [-360, 360]

# Имена деталей
ARMORX_PART = ['Bracket', 'LX-16A', 'Keeper_1', 'Keeper_2', 'Flange', 'Holder',
'Linker', 'CrossBar', 'CrossFlange', 'Corner']
# Имена сборок
ARMORX_ASSEM = ['Box', 'Base_link', 'A1_link', 'A2_link', 'A3_link', 'A4_link',
'ArmorX']

#**********************************
# Класс анимации манипулятора ArmorX
#
# delay - Время между кадрами анимации (сек)
# duration - Общее время выполнения анимации (сек)
#**********************************
class ArmorX( Thread):
    def __init__( self, delay, duration):
        super( ArmorX, self).__init__()

        self.name = type( self).__name__
        self.cnt = 0
        self.maxcnt = round( duration / delay)
        self.delay = delay

        doc_list = App.listDocuments()
        for doc in ARMORX_ASSEM:
            if doc in doc_list:
                continue
            App.openDocument( "%s.FCStd" % doc)

        App.setActiveDocument( "ArmorX")
        self.Doc = App.ActiveDocument

        self.Variables = self.Doc.getObject( "Variables")
        self.Model = self.Doc.getObject( "Model")

        Base_link = self.Doc.getObject( "Base_link")
        A1_link = self.Doc.getObject( "A1_link")
        A2_link = self.Doc.getObject( "A2_link")
        A3_link = self.Doc.getObject( "A3_link")
        A4_link = self.Doc.getObject( "A4_link")

        self.Base_var = Base_link.LinkedObject.getObject( "Variables")
        self.A1_var = A1_link.LinkedObject.getObject( "Variables")
        self.A2_var = A2_link.LinkedObject.getObject( "Variables")
        self.A3_var = A3_link.LinkedObject.getObject( "Variables")
        self.A4_var = A4_link.LinkedObject.getObject( "Variables")

        # Текущий угол сочленения
        self.a1_joint = self.Base_var.A1_joint
        self.a2_joint = self.A1_var.A2_joint
        self.a3_joint = self.A3_var.A3_joint
        self.a4_joint = self.A3_var.A4_joint
        self.a5_joint = self.A4_var.A5_joint

        # Направление вращения сочленения
        self.a1_turn = True
        self.a2_turn = True
        self.a3_turn = True
        self.a4_turn = True
```

```
        self.a5_turn = True

        if self.a1_joint < A1_LIMIT[0]:
            self.a1_joint = A1_LIMIT[0]
        elif self.a1_joint >= A1_LIMIT[1]:
            self.a1_joint = A1_LIMIT[1] - 1

        if self.a2_joint < A2_LIMIT[0]:
            self.a2_joint = A2_LIMIT[0]
        elif self.a2_joint >= A2_LIMIT[1]:
            self.a2_joint = A2_LIMIT[1] - 1

        if self.a3_joint < A3_LIMIT[0]:
            self.a3_joint = A3_LIMIT[0]
        elif self.a3_joint >= A3_LIMIT[1]:
            self.a3_joint = A3_LIMIT[1] - 1

        if self.a4_joint < A4_LIMIT[0]:
            self.a4_joint = A4_LIMIT[0]
        elif self.a4_joint >= A4_LIMIT[1]:
            self.a4_joint = A4_LIMIT[1] - 1

        if self.a5_joint < A5_LIMIT[0]:
            self.a5_joint = A5_LIMIT[0]
        elif self.a5_joint >= A5_LIMIT[1]:
            self.a5_joint = A5_LIMIT[1] - 1

        print( "(%s) delay=%.2f" % (self.name, self.delay))
        print( "(%s) A1_joint=%.2f" % (self.name, self.a1_joint))
        print( "(%s) A2_joint=%.2f" % (self.name, self.a2_joint))
        print( "(%s) A3_joint=%.2f" % (self.name, self.a3_joint))
        print( "(%s) A4_joint=%.2f" % (self.name, self.a4_joint))
        print( "(%s) A5_joint=%.2f" % (self.name, self.a5_joint))

        # Инициализация начального положения сочленений
        self.Base_var.A1_joint = self.a1_joint
        self.A1_var.A2_joint = self.a2_joint
        self.A3_var.A3_joint = self.a3_joint
        self.A3_var.A4_joint = self.a4_joint
        self.A4_var.A5_joint = self.a5_joint

    def run( self):
        while True:
            time.sleep( self.delay)
            self.cnt += 1
            if self.cnt > self.maxcnt:
                print( "(%s).run: exit" % self.name)

#                for doc in ARMORX_ASSEM[::-1]:
#                    App.closeDocument( doc)
#                for doc in ARMORX_PART[::-1]:
#                    App.closeDocument( doc)

                return;

            print( "(%s).run: cnt=%d [%d]" % (self.name, self.cnt, self.maxcnt))
            self.animate()

    # Анимация сочленений
    def animate( self):
        print( "(%s).animate[%d]:" % (self.name, self.cnt))
```

```
        if not ( self.a1_joint >= A1_LIMIT[0] and self.a1_joint < A1_LIMIT[1]):
            self.a1_turn = not self.a1_turn
        self.a1_joint += A1_STEP if self.a1_turn else -A1_STEP

        if not ( self.a2_joint >= A2_LIMIT[0] and self.a2_joint < A2_LIMIT[1]):
            self.a2_turn = not self.a2_turn
        self.a2_joint += A2_STEP if self.a2_turn else -A2_STEP

        if not ( self.a3_joint >= A3_LIMIT[0] and self.a3_joint < A3_LIMIT[1]):
            self.a3_turn = not self.a3_turn
        self.a3_joint += A3_STEP if self.a3_turn else -A3_STEP

        if not ( self.a4_joint >= A4_LIMIT[0] and self.a4_joint < A4_LIMIT[1]):
            self.a4_turn = not self.a4_turn
        self.a4_joint += A4_STEP if self.a4_turn else -A4_STEP

        if not ( self.a5_joint >= A5_LIMIT[0] and self.a5_joint < A5_LIMIT[1]):
            self.a5_turn = not self.a5_turn
        self.a5_joint += A5_STEP if self.a5_turn else -A5_STEP

#        self.Base_var.A1_joint = self.a1_joint
#        self.A1_var.A2_joint = self.a2_joint
#        self.Doc.recompute()
        setattr( self.Base_var, "A1_joint", self.a1_joint)
        setattr( self.A1_var, "A2_joint", self.a2_joint)
        setattr( self.A3_var, "A3_joint", self.a3_joint)
        setattr( self.A3_var, "A4_joint", self.a4_joint)
        setattr( self.A4_var, "A5_joint", self.a5_joint)
        self.Doc.Model.recompute( "True")
        Gui.updateGui()

        print( "(%s).animate[%d]: a1=%.2f, a2=%.2f, a3=%.2f, a4=%.2f, a5=%.2f" %
                (self.name, self.cnt, self.a1_joint, self.a2_joint,
self.a3_joint, self.a4_joint, self.a5_joint))

# Старт анимации
def start( delay, duration):
    print( "(armorx) start: delay=%.2f, duration=%.2f" % (delay, duration))

    armorx = ArmorX( delay, duration)
    armorx.start()
# Нельзя: Блокирует главный поток FreeCAD!!!
#    armorx.join()

    print( "(armorx) start: end")

start( 0.2, 100.0)
```

Список литературы

1. **A FreeCAD manual**, Yorik Van Havrik and the FreeCAD Community.
2. **A Sketcher Lecture**, Christoph Blaue, May 13, 2020.