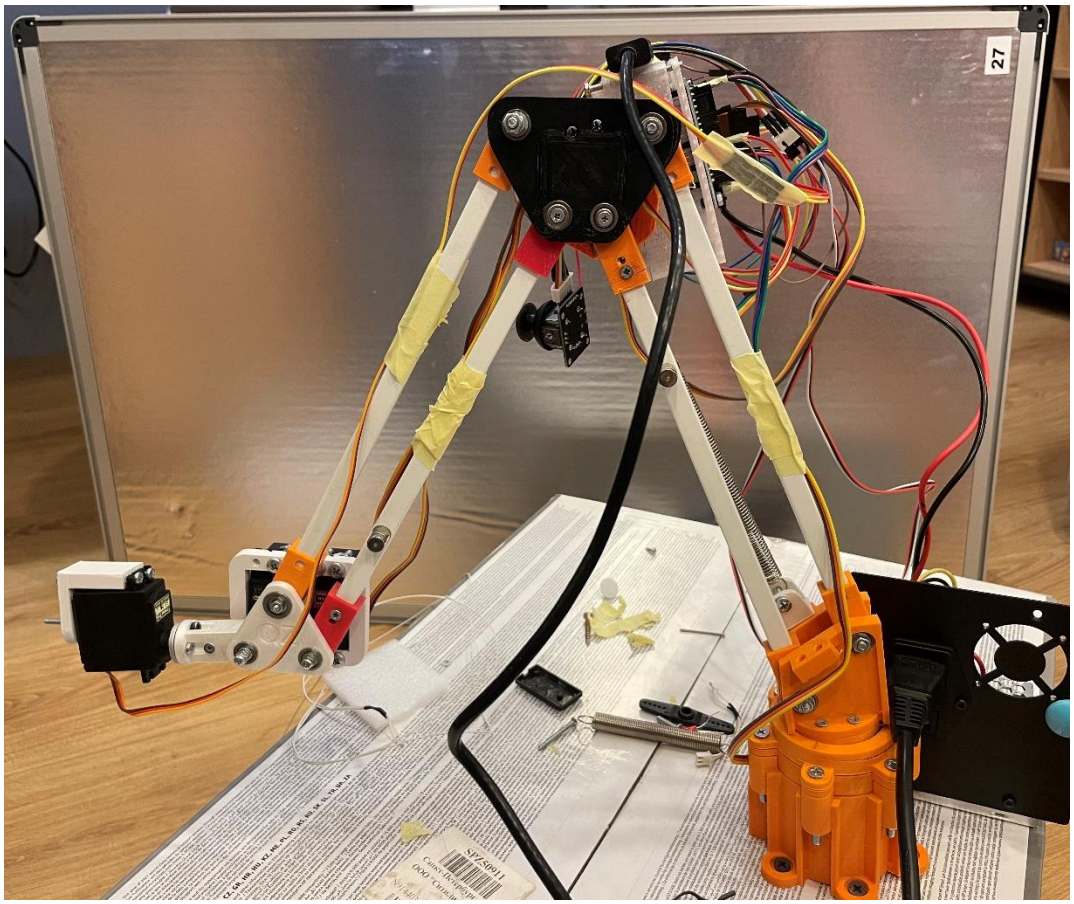
# Отчет по проекту «Tertiarm-ROS»

Выполнил студент гр. 3331506/80401 Пантелеев М. Д.
Преподаватель Хазанский Р.Р.

В качестве своего курсового проекта я заявлял пятиосевой манипулятор. Что ж, его я и собрал. Behold!




Видео работы: [ссылка]

Да уж, выбрать ракурс, с которого оно выглядит более-менее прилично не так-то просто.

## Предисловие

Собрать и запрограммировать себе этот манипулятор я хотел ещё году в 2017-18. Идея была в том, чтобы дистанционно снимать с 3Д-принтера напечатанные детальки. Тогда я наткнулся на проект мейкера Karagad, на основании которого и начал собирать свою руку. В конструкции понравилось то, что, будучи основанной на лампе от икеи, она может поддерживать своё положение за счёт пружин, не сильно нагружая двигатели. Однако в те годы знаний, чтобы управлять рукой кроме как последовательным заданием углов вручную, не хватило. Теперь же благодаря трём годам обучения и изучению STM появилась возможность наконец закончить старый проект.

## Ход работы

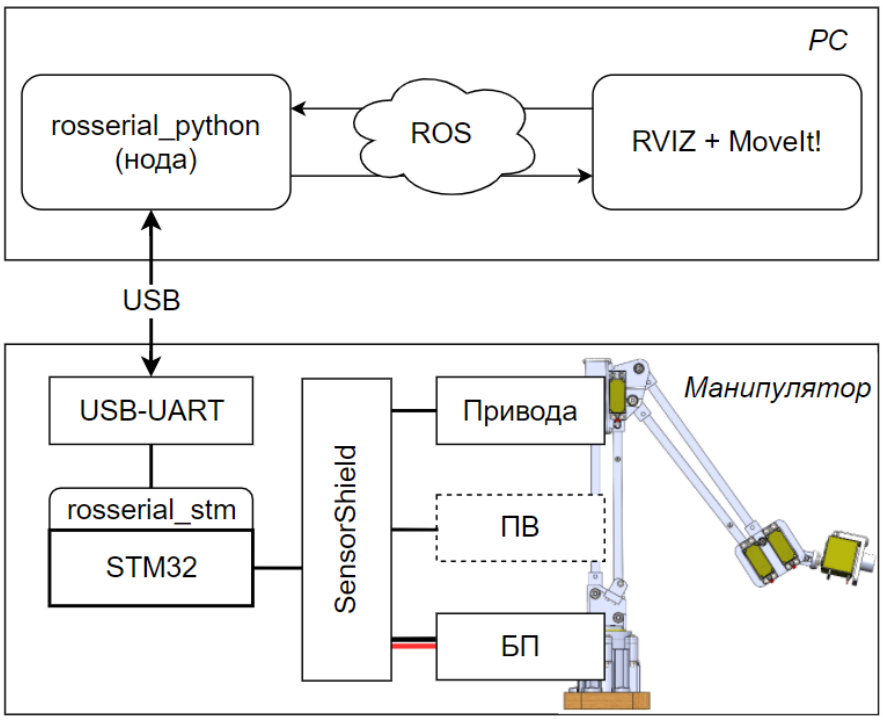
Идею делать сначала максимально простое управление через терминал, изложенную в начальном ТЗ, я отбросил почти сразу и взялся за rosserial. Оказалось, что этой библиотеки для семейства F1 нет, и чтобы ей пользоваться, надо адаптировать под свой процессор. Rosserial опирается на HAL, что в общем-то и определило выбор периферийной библиотеки.

Дальнейшие трудности в основном не были связаны с МК. Надо было много ~~бороться~~ работать с линуксом и росом, искать точную ширину импульса для каждой сервы на глаз, перепечатывать и пересобрать некоторые узлы.

Какое-то время не получалось привести руку в движение, хотя точно была связь, и соответствующая функция выполнялась. Полностью в том, что (не) происходило, я не разобрался, но повышение бодрейта до 115200 решило проблему.

## Что и как тут работает

Манипулятор управляется через ROS. В Rviz находится виртуальная модель робота, оператор или специальная нода (например, для управления с помощью геймпада) могут обновлять целевое состояние руки. По команде MoveIt! просчитывает обратную кинематику и публикует в специальный топик значения углов. Rosserial_python делает этот топик доступным для прослушивания микроконтроллером, который подключен по UART через USB-переходник. На самой же stm-ке на каждую «посылку» срабатывает функция, переводящая полученные углы в понятный вид с учётом различий модели и реального робота и скармливающая их сервоприводам. В режиме ручного управления оператор с помощью джойстика может перемещать звенья в нужные ему положения.




Структурная схема (ПВ – Пульт выносной, БП – Блок питания 5В)

## Обзор структуры проекта

Код в отчёт я вставлять не буду, опишу только назначение основных файлов и функций.

В main.cpp (плюсы нужны для роса) содержатся константы для расчёта углов двигателей, объекты и callback'и роса и функции инициализации периферии. В главном цикле while есть два участка – для ручного и удалённого управления.

В проекте есть папка mylibs, в которой содержатся файлы, обеспечивающие конфигурацию и взаимодействие с двигателями, джойстиком и кнопками.

- /SERVO: servo_cfg.c – описание экземпляров сервоприводов; servo.c – инициализация всех экземпляров и управление приводами.
- /JOYSTICK: joystick.c – инициализация и чтение каналов ADC.
- /UTILS: DWT_delay.c – задержки с использованием DWT; button.c – функция для фонового обновления статуса кнопки в прерывании; filter.c – борьба с дребезгом с помощью суммирующего буфера.

## Вещи, которые хотелось бы доработать

Несмотря на то, что я достиг всего, на что рассчитывал в начале семестра, я всё ещё вижу много направлений, по которым проект можно сделать лучше. Вот некоторые из них:

- Дёргания при движении. Связаны с тем, что ROS выдаёт значения с некоторым шагом. Можно экспериментировать с частотами публикации и считывания или продумать интерполяцию.
- Точность и повторяемость. Нужно более точно искать базы, граничные значения ширины импульса, улучшать модель.
- Однажды в процессе настройки одна из серв **расплавилась**. Это было неприятно и натолкнуло на размышления о системе безопасности, которая бы отрубала движки при возрастании потребляемого тока или температуре на двигателях.
- Ручное управление. Контроль джойстиком с запоминанием точек и последующим их воспроизведением. Как я писал, этот пункт сейчас в работе. Немного напрягает количество проводов, которое надо тянуть на джойстик да пару кнопок, но, с другой стороны, это повод попробовать радио или какую-нибудь плату с цифровым интерфейсом.
- Дальнейшая интеграция с ROS.

## Рефлексия по итогу

Несмотря на вышеперечисленное, проектом я доволен. То, что у меня получилось, можно считать платформой, на которой я могу дальше изучать STM и ROS, внедряя новые функции. К тому же свой первый bluepill я купил только в этом году, так что это ещё и мой первый проект на stm, получается.

С точки зрения изучения программирования микропроцессоров, мне кажется, я узнал не так много, как мои одногруппники, делающие лабы. В определённый момент это даже заставило задуматься о смене проекта на другой, более навороченный в плане использования различной периферии (могу подсказать как пример для будущих поколений). К тому же защита лаб позволяет получить какой-то фидбек по коду и правильной реализации заданий. Возможно, я поделал бы лабы летом, чтобы охватить темы, которых я не коснулся в проекте и глубже изучить вещи, которые за меня делала библиотека роса, например.

С точки зрения изучения ROS я узнал много всего, но в подробности вдаваться не буду, наверное, потому что это не по теме.

Также я связался с автором оригинальной руки, узнал, что это парень из России, который сейчас работает над второй версией в составе набора для обучения детей робототехнике. Возможно, получится в этой области посотрудничать или просто внести вклад в его проект.

Как бы то ни было, спасибо за курс!