Arm Controller

Принцип работы системы

Принцип работы системы. на рис.1

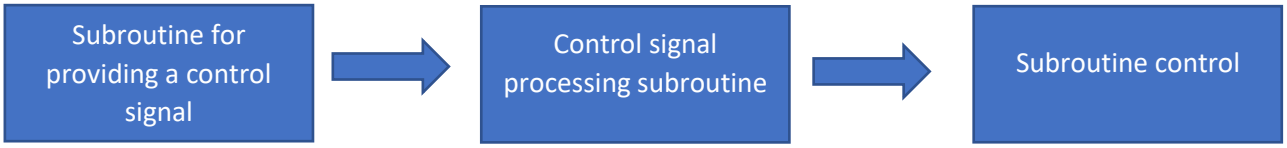

Рис.1. Принцип работы системы

1. На подпрограмму подачи управляющего сигнала приходит изображение из двух камер (углы камер настраиваются, но оптимальный вариант для возможности наблюдать за сервомотором на рис.2). Данная подпрограмма должна обрабатывать полученные изображения и генерировать управляющий сигнал для манипулятора который будет отправляться с помощью библеотеки ArmControl.

2. Подпрограмма обработки сигналов управления работает по принципу обработки сокетов.

3. Подпрограмма управления отвечает за получение информации и их физического выполнения.

Быстрый старт

Вместе с инструкцией прикреплен файл **Client/source/ArmCamera.py** для проверки работоспособности системы, в нем есть вывод информации с камер с возможностью управлять рукой в режиме реального времени с помощью графического интерфейса (рис.2.). Для корректной работы системы необходимо открыть файл **camera.cnf** и в нем прописать идентификаторы камер в системе. Данный пример демонстрирует работу с библиотекой **ArmControl** для дальнейшего управления манипулятором. Данную подпрограмму рекомендуется заменить на другую з автоматическим расчётом движения манипулятора.

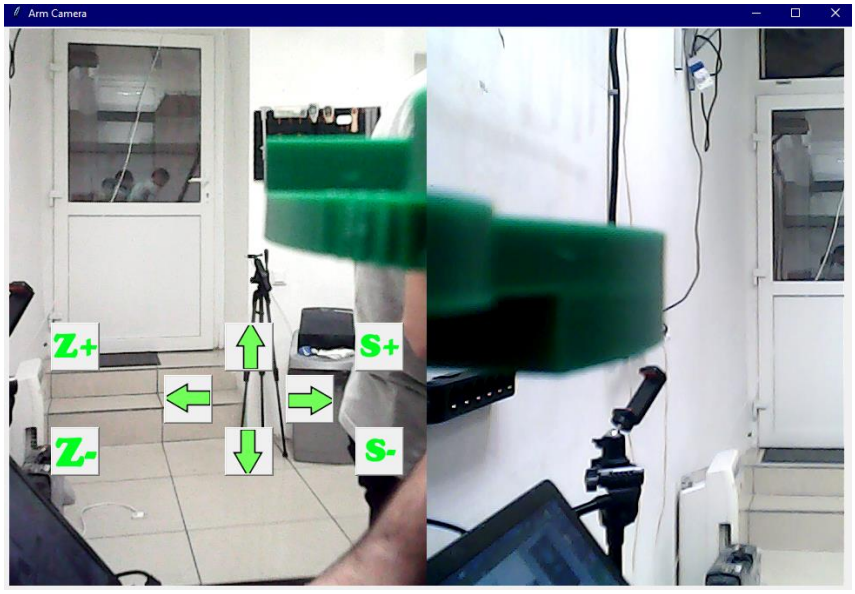

Рис.2. Пример интерфейса **ArmCamera**

Подпрограмма обработки сигналов управления

Подпрограмма обработки сигналов управления **Server/release/TCPServer.exe** есть обязательной подпрограммой, вносить изменения в которую крайне ненужно. Рекомендуется только произвести ее настройку при запуске. А именно выбрать порт для общения с рукой.

Исходный код программы можно посмотреть по адресу: **Server/source/**

Пример полного запуска системы

1. Подключаем манипулятор к компьютеру с проверяем COM-порты.
2. Подключаем камеры к компьютеру. Указываем идентификаторы камер в файле **camera.cnf**
3. Запускаем **Server/release/TCPServer.exe**
4. Запускаем **Client/source/ArmCamera.py**

Подпрограмма подачи управляющего сигнала

Для интеграции Подпрограмма подачи управляющего сигнала в ваше приложение достаточно сделать следующее

1. Подключить библиотеку

   import ArmControl

2. Визвать нужную вам функцию для управления рукой. Например функцию ***setZero()***

   ArmControl.setZero()

3. Все готово!)

   Перечень доступних команд:
   - *setZero()* – установить манипулятор в нулевую позицию
   - *setCordinates(x,y,z)* – задать координаты ***x,y,z***
   - *getCordinates()* – получить позицию манипулятора
   - *setXCordinates(x)* – установить позицию манипулятора в ***x***
   - *setYCordinates(y)* – установить позицию манипулятора в ***y***
   - *setZCordinates(z)* – установить позицию манипулятора в ***z***
   - *getXCordinates()* – получить текущую координату ***x***
   - *getYCordinates()* – получить текущую координату ***y***
   - *getZCordinates()* – получить текущую координату ***z***
   - *getServo()* – получить текущий угол серводвигателя
   - *setServo(x)* – установить серводвигатель в позицию ***x***
   - *ServoAddAngle()* – прибавить к текущему углу серводвигателя **10** градусов
   - *ServoAddAngle(angle)* – прибавить к текущему углу серводвигателя ***angle*** градусов
   - *ServoSubtractAngle()* – отнять от текущего угла серводвигателя ***10*** градусов
   - *ServoSubtractAngle(angle)* – отнять от текущего угла серводвигателя ***angle*** градусов
   - *ZCordinatesAdd()* – прибавить к текущей z координате ***1*** градус
   - *ZCordinatesSubtract()* – отнять от текущей z координати ***1*** градус
   - *YCordinatesAdd()* – прибавить к текущей y координате ***1*** градус
   - *YCordinatesSubtract()* – отнять от текущей y координати ***1*** градус
   - *XCordinatesAdd()* – прибавить к текущей x координате ***1*** градус
   - *XCordinatesSubtract()* – отнять от текущей x координати ***1*** градус