ЭКЗОСКЕЛЕТОННЫЙ РОБОТИЗИРОВАННЫЙ КОМПЛЕКС

Программа управления электроприводом

Описание применения

БЛКУ.00291-01 31 180

Листов 21

Инв. № подл.	Подп. и дата.	Взам. инв. №	Инв. № дубл.	Подп. и дата.

2015

Аннотация

В данном программном документе приведено описание применения компонента "Программа управления электроприводом" (далее Программа), который предназначен для обеспечения доступа прикладным программам встроенной системы управления к приводам и датчикам состояния экзоскелетонного роботизированного комплекса (далее ЭРК).

Раздел «Назначение программы» содержит описание назначения программы, возможности данной программы, а также ее основные характеристики и ограничения, накладываемые на область применения программы.

В разделе «Условия применения» указаны условия, необходимые для выполнения программы (требования к необходимым для данной программы техническим средствам, и другим программам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера).

Раздел «Описание задачи» посвящен задачам, решаемым с помощью программы, указаны определения задач и методы их решения.

В разделе «Входные и выходные данные» указаны сведения о входных и выходных данных.

Оформление программного документа «Описание программы» произведено в соответствии с требованиям ЕСПД ГОСТ 19.502-78.

Содержание

Глоссарий

АЦП Аналогово-цифровой преобразователь

ПОС Плата оцифровки и сопряжения

ЭРК Экзоскелетонный роботизированный комплекс

1 Назначение программы

Основное назначение компонента "Программа управления электроприводом" (шифр CBLSD): обеспечение доступа к электроприводам и датчикам экзоскелетонного роботизированного комплекса (ЭРК).

В качестве электроприводов в рамках ЭРК применяются бесколлекторные электродвигатели с датчиками Холла, управляемые посредством специализированных контроллеров BLSD-20, подключаемых к микропроцессорной плате BeagleBoard XM с помощью преобразователя SC&T RS0001I-2. В составе ЭРК 2 бесколлекторных электропривода Maxon EC 45 flat 70 W 36 V, 2 бесколлекторных электропривода Maxon EC 90 flat 90 W 36 V и 4 контроллера BLSD-20 (по одному контроллеру на каждый электропривод).

В качестве датчиков применяются магнитные инклинометры ASM PRAS21 с магнитом ACM PMAG22. В состав ЭРК входят 6 датчиков - по одному в каждом суставе нижних конечностей. Датчики подключаются к плате оцифровки и сопряжения (далее ПОС), которая, в свою очередь, подключается к вычислительной платформе BeagleBoard XM rev. C.

2 Условия применения

2.1 Платформа

Программный компонент CBLSD функционирует на платформе BeagleBoard XM rev. C под управлением операционной системы Angstrom 2011.3, Dureza (ядро Linux 2.6.32).

2.2 Подсистема датчиков

К микропроцессорной плате BeagleBoard XM должна быть подключена плата ПОС, к которой, в свою очередь, должны быть подключены: 6 датчиков ASM PRAS21 и один МЭМС-сенсор Analog Devices ADIS16407BMLZ. Схема подключения датчиков, определяющих вектор состояния биомеханической системы ЭРК-пилот приведена на рисунке 2.1.

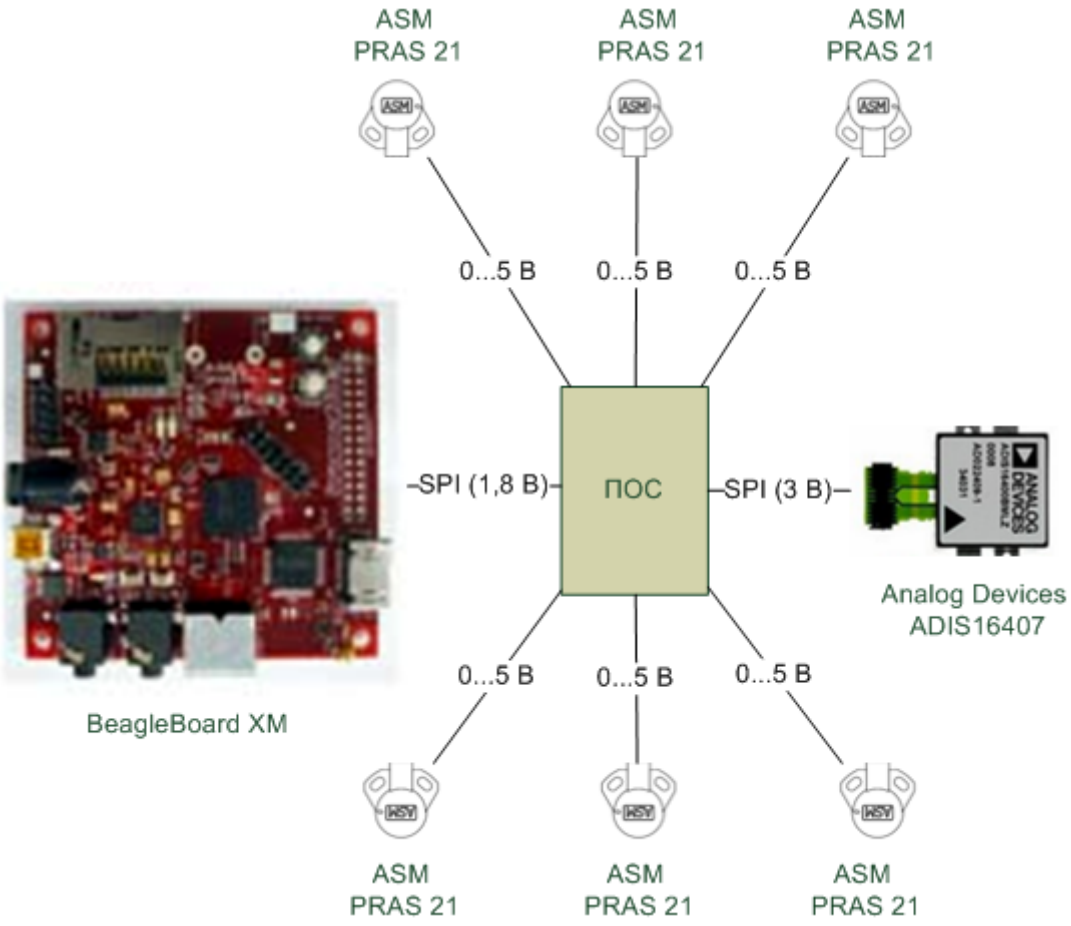

Рисунок 2.1 - Общая схема подключения датчиков состояния к микропроцессорной плате BeagleBoard XM

3 Описание задачи

3.1 Организация взаимодействия с программой

Программа имеет программный интерфейс, обеспечивающий доступ к функциональности компонента посредством заголовочных файлов:

- group_of_engines_API.h;
- nav_sensors_API.h

оформленных на языке C++.
Для организации взаимодействия с программой необходимо:

- включить в код разрабатываемого приложения (библиотеки) директивы:

```
#include "group_of_engines_API.h";
#include "nav_sensors_API.h";
```

- линковщику указать зависимость разрабатываемого приложения (библиотеки) от динамических библиотек libgroup_of_engines_API.so и libnav_sensors_API.so.

3.2 Создание инфраструктуры управления приводами

Под инфраструктурой управления приводами в данном документе понимается набор программных объектов, обеспечивающих синхронное управление комплектом электроприводов. Перед созданием инфраструктуры необходимо выяснить:

- количество приводов в группе;
- адреса приводов на шине RS485;
- COM-порт, к которому подключен преобразователь RS232-RS485;
- полные пути до файлов с таблицами калибровки приводов.

В первую очередь, необходимо создать объекты, ассоциированные с конкретными приводами (объекты класса Engine) и скомпоновать их в комплект (объект класса GroupOfEngines).

Пример 3.2.1:

```
#include "group_of_engines_API.h";
#include "nav_sensors_API.h";

using namespace Blsd;

//Создание объектов - приводов
Engine left_thigh(LeftThighEngineID, "left thigh", LeftThighEngineID,
                              "/opt/erc/left_thigh_calibration_table.cnfg");
Engine left_knee(LeftKneeEngineID, "left knee", LeftKneeEngineID,
                              "/opt/erc/left_knee_calibration_table.cnfg");
Engine right_thigh(RightThighEngineID, "right thigh", RightThighEngineID,
                               "/opt/erc/right_thigh_calibration_table.cnfg");
Engine right_knee(RightKneeEngineID, "right knee", RightKneeEngineID,
                              "/opt/erc/right_knee_calibration_table.cnfg");
```

```
GroupOfEngines erk("/dev/ttyUSB0", ELECTROPRIVOD);//COM-порт интерфейса RS232-RS485
//и протокол управления контроллером привода
//GroupOfEngines erk("/dev/ttyS2", NIFTI);
    //Компоновка приводов в группу
erk.AddEngineToGroup(left_thigh);
erk.AddEngineToGroup(left_knee);
erk.AddEngineToGroup(right_thigh);
erk.AddEngineToGroup(right_knee);
```

3.3 Выполнение калибровки приводов

Для выполнения калибровки сустава необходимо заполнить и сохранить в файле таблицу, сопоставляющую момент, развиваемый приводом, и код АЦП датчика тока. Для этого:

1. формируют список кодов АЦП датчика тока, для которых будут проводиться измерения;
2. создают объекты типов Engine (привод) и GroupOfEngines (группа приводов);
3. вызывают метод GroupOfEngines::AddEngineToGroup(), который добавляет привод в группу (объект типа Engine передается как параметр в метод GroupOfEngines::AddEngineToGroup());
4. для каждого кода АЦП из списка:
    - вызывают метод GroupOfEngines::SetADC(), который задает значение кода АЦП для привода (идентификатор привода в группе (индекс в массиве приводов) передается как первый параметр в метод GroupOfEngines::SetADC(), устанавливаемое значение кода АЦП передается как второй параметр);
    - вызывают метод GroupOfEngines::SendMomentsForAllEngines(), который устанавливает заданные коды АЦП для каждого привода в группе;
    - запрашивают у оператора текущее значение момента (в Нм), зафиксированного с помощью динамометра;
    - после ввода значения оператором вызывают метод CalibrateTable::FillTable(), который заполняет калибровочную таблицу (устанавливаемое значение кода АЦП передается как первый параметр CalibrateTable::FillTable(), момент, развиваемый приводом, как второй параметр);
5. После обработки всего списка кодов АЦП датчика тока сохраняют калибровочную таблицу в файле, вызвав метод CalibrateTable::FlashToFile().

Пример 3.3.1:

```
//...
//коды АЦП, для которых выполняется калибровка
unsigned int dutyCycles[ADCCODE_QUAN]={ 0, 2, 3, 4, 7, 9, 10, 12, 14, 16 };

//создаем объект - привод
Engine ngn(LeftThighEngineID, "left thigh", LeftThighEngineID, "");

GroupOfEngines grp("/dev/ttyUSB0", ELECTROPRIVOD);//создаем группу приводов
grp.AddEngineToGroup(ngn);//добавляем привод в группу

for (unsigned int i = 0; i < ADCCODE_QUAN; i++) //итерируемся по всем заданным кодам АЦП
```

```
{
    grp.SetADC(LeftThighEngineID, adcCodes[i]);//задаем значение кода АЦП для привода
    grp.SendMomentsForAllEngines();//устанавливаем значение кода АЦП

    //запрашиваем значение момента, развиваемого в суставе
    double moment = 0;
    cout << "Specify the current moment, Nm: " << endl;
    cin >> moment;

    ngn._cal_table.FillTable(adcCodes[i],moment);//заполняем калибровочную таблицу
}

//сохраняем таблицу в файл
ngn._cal_table.FlashToFile("/opt/erc/left_thigh_calibration_table.cnfg");
//...
```

3.4 Управление счетчиком оборотов привода

Для получения значения количества оборотов привода необходимо вызвать метод GroupOfEngines::GetRounds(). Чтобы сбросить счетчик оборотов привода необходимо вызвать метод GroupOfEngines::ResetRoundsCounter().

Пример 3.4.1:

```
#include "group_of_engines_API.h";
#include "nav_sensors_API.h";

using namespace Blsd;

//Создание объектов - приводов
Engine left_thigh(LeftThighEngineID, "left thigh", LeftThighEngineID,
                            "/opt/erc/left_thigh_calibration_table.cnfg");

GroupOfEngines erk("/dev/ttyUSB0", ELECTROPRIVOD);//COM-порт интерфейса RS232-RS485
//и протокол управления контроллером привода

//Компоновка приводов в группу
erk.AddEngineToGroup(left_thigh);

erk.SendMomentsForAllEngines();//Одновременная выдача заданных моментов всеми приводами

//Получение количества оборотов привода
short rounds = erk.GetRounds(LeftThighEngineID);//идентификатор привода

//Сбросить счетчик оборотов привода
erk.ResetRoundsCounter(LeftThighEngineID);//идентификатор привода
```

3.5 Получение угловой скорости привода

Для получения угловой скорости привода необходимо перед отправкой групповой команды установить режим приема ответов от контроллера. Для этого необходимо выставить флаг GroupOfEngines::isBLSDResponce в значение true. Далее нужно вызвать метод GroupOfEngines::GetAngleSpeed().

Пример 3.5.1:

```
#include "group_of_engines_API.h";
#include "nav_sensors_API.h";

using namespace Blsd;
```

```
//Создание объектов - приводов
Engine left_thigh(LeftThighEngineID, "left thigh", LeftThighEngineID,
                            "/opt/erc/left_thigh_calibration_table.cnfg");

GroupOfEngines erk("/dev/ttyUSB0", ELECTROPRIVOD);//COM-порт интерфейса RS232-RS485
//и протокол управления контроллером привода

//Компоновка приводов в группу
erk.AddEngineToGroup(left_thigh);

erk.isBLSDResponce = true;//Устанавливаем режим приема ответов от контроллера

erk.SendMomentsForAllEngines();//Одновременная выдача заданных моментов всеми приводами

//Получение угловой скорости привода
int angle_speed = erk.GetAngleSpeed(LeftThighEngineID);//идентификатор привода
```

3.6 Получение количества оборотов привода для калибровки

Для получения количества оборотов привода для калибровки необходимо перед отправкой групповой команды установить режим приема ответов от контроллера. Для этого необходимо выставить флаг GroupOfEngines::isBLSDResponce в значение true. Далее нужно вызвать метод GroupOfEngines::GetAngleSpeed().

Пример 3.5.1:

```
#include "group_of_engines_API.h";
#include "nav_sensors_API.h";

using namespace Blsd;

//Создание объектов - приводов
Engine left_thigh(LeftThighEngineID, "left thigh", LeftThighEngineID,
                            "/opt/erc/left_thigh_calibration_table.cnfg");

GroupOfEngines erk("/dev/ttyUSB0", ELECTROPRIVOD);//COM-порт интерфейса RS232-RS485
//и протокол управления контроллером привода

//Компоновка приводов в группу
erk.AddEngineToGroup(left_thigh);

erk.isBLSDResponce = true;//Устанавливаем режим приема ответов от контроллера

erk.SendMomentsForAllEngines();//Одновременная выдача заданных моментов всеми приводами

//Получение количества оборотов привода для калибровки
int num_rev = erk.GetNumRevCalibration(LeftThighEngineID);//идентификатор привода
```

3.7 Управление моментами приводов

Задача управления моментами приводов решается в два шага. Сначала необходимо задать желаемые значения моментов посредством метода GroupOfEngines::SetMoment(), далее необходимо одной командой установить заданные моменты у всех приводов группы (GroupOfEngines::SendMomentsForAllEngines()).

Пример 3.7.1:

```
#include "group_of_engines_API.h";
#include "nav_sensors_API.h";

using namespace Blsd;
```

```
//Создание объектов - приводов
Engine left_thigh(LeftThighEngineID, "left thigh", LeftThighEngineID,
                            "/opt/erc/left_thigh_calibration_table.cnfg");

GroupOfEngines erk("/dev/ttyUSB0", ELECTROPRIVOD);//COM-порт интерфейса RS232-RS485
//и протокол управления контроллером привода

//Компоновка приводов в группу
erk.AddEngineToGroup(left_thigh);

SetMoment(LeftThighEngineID, 10);//Задание момента для конкретного привода


erk.SendMomentsForAllEngines();//Одновременная выдача заданных моментов всеми приводами
```

3.8 Блокировка/разблокировка вала привода

Задача блокировки выльв привода решается в два шага. Сначала необходимо вызвать метод GroupOfEngines::LockEngine(), далее необходимо одной командой установить заданные моменты у всех приводов группы (GroupOfEngines::SendMomentsForAllEngines()). Операция разблокировки выполняется аналогично, но вызывается метод UnlockEngine().

Пример 3.8.1:

```
#include "group_of_engines_API.h";
#include "nav_sensors_API.h";

using namespace Blsd;

//Создание объектов - приводов
Engine left_thigh(LeftThighEngineID, "left thigh", LeftThighEngineID,
                            "/opt/erc/left_thigh_calibration_table.cnfg");

GroupOfEngines erk("/dev/ttyUSB0", ELECTROPRIVOD);//COM-порт интерфейса RS232-RS485
//и протокол управления контроллером привода

//Компоновка приводов в группу
erk.AddEngineToGroup(left_thigh);

//Блокировка вала конкретного привода. Для разблокировки UnlockEngine(LeftThighEngineID)
erk.LockEngine(LeftThighEngineID);


erk.SendMomentsForAllEngines();//Одновременная выдача заданных моментов всеми приводами
```

3.9 Выполнение калибровки сенсоров-инклинометров

Плата оцифровки и сопряжения ПОС, к которой подключается датчики-инклинометры, имеет на борту 8-ми канальный АЦП. К шести входам (0-5) АЦП подключаются шесть инклинометров:

- вход 0 - инклинометр левого тазобедренного сустава;
- вход 1 - инклинометр левого коленного сустава;
- вход 2 - инклинометр левого голеностопного сустава;
- вход 3 - инклинометр правого тазобедренного сустава;
- вход 4 - инклинометр правого коленного сустава;

- вход 5 - инклинометр правого голеностопного сустава.

Для выполнения калибровки сустава необходимо заполнить и сохранить в файле градуировочную характеристику, сопоставляющую коду АЦП угол, под которым согнут сустав. Для этого:

1. формируют список углов, для которых будут проводиться измерения;
2. создают объекты типа CalibrationTable (хранилище пар "код АЦП-угол");
3. получают интерфейс типа NavSensors посредством вызова глобальной функции GetNavSensors();
4. активируют соединение с навигационными датчиками посредством вызова у интерфейса NavSensors метода Connect() с указанием необходимой частоты обновления показаний датчиков (800 - 1000 Гц) и ширины окна медианного фильтра (3 отсчета);
5. для каждого угла из списка:
   - выдают оператору сообщение о необходимости ручной установки заданного угла (в градусах);
   - запрашивают у оператора значение установленного угла (в градусах), измеренное с помощью угломерного инструмента;
   - после ввода значения оператором вызывают метод NavSensors::GetData(), который возвращает структуру типа NavSensorsData. Поле NavSensorsData::inclinometer_adc_code представляет собой массив кодов АЦП, зафиксированных для шести входов ПОС (0 - 5);
   - из массива NavSensorsData::inclinometer_adc_code извлекают код АЦП калибруемого канала (по индексу входа);
   - пару "код АЦП-угол", записывают в градуировочную таблицу, вызвав метод CalibrationTable::AddRelation().
6. После обработки всего списка углов сохраняют калибровочную таблицу в файле, вызвав метод CalibrationTable::FlashToFile().

Пример 3.9.1:

```
//...

CalibrationTable cal_table; //градуировочная характеристика датчика-инклинометра

NavSensors * POS = GetNavSensors(); //интерфейс системы сенсоров

//углы, для которых выполняется калибровка (градусы)
double input_angles[ANGLES_QUAN]={0, 1, 2, 20, 40, 60, 80, 90, 100, 107, 108, 109, 110};

bool connected = POS -> Connect(200, 3);//активируем соединение с датчиками, частота опроса - 200 Гц
if(!connected) // если функция вернула false, то не удалось подключится
{
      std::cout << "Failed to establish connection with sensors" << std::endl;
      NavSensorsError error = POS -> GetLastError(); //получаем код ошибки
      std::cout << GetErrorDescription(error) << std::endl; //выводим описание ошибки
}

for(unsigned int i=0; i<ANGLES_QUAN; i++) //итерируемся по всем заданным углам
{
      //Запрос ручной установки угла сгибания сустава.
```

```
        std::cout<<"Set angle of joint as: " << input_angles[i] << " degrees." <<std::endl;
        std::cout<< "Input actual angle (degrees): ";

        double actual_angle=0; //актуальное значение угла (градусы)
        std::cin >> actual_angle; //ввод актуального значения угла оператором

        //фиксируем текущие показания всех сенсоров, подключенных к ПОС
        NavSensorsData curr_state=POS -> GetData(); //фиксируем текущие показания всех сенсоров,

        //дополняем градуировочную характеристику для левого бедра
        cal_table.AddRelation(curr_state.inclinometer_adc_code[LeftThighSensorID], actual_angle);
    }

cal_table.FlashToFile("home/root/RightThighSensor.clb") ; //сохраняем таблицу в файл
//...
```

3.10 Получение данных о состоянии ЭРК

Плата оцифровки и сопряжения ПОС, к которой подключается датчики-инклинометры, имеет на борту 8-ми канальный АЦП. К шести входам (0-5) АЦП подключаются шесть инклинометров:

- вход 0 - инклинометр левого тазобедренного сустава;
- вход 1 - инклинометр левого коленного сустава;
- вход 2 - инклинометр левого голеностопного сустава;
- вход 3 - инклинометр правого тазобедренного сустава;
- вход 4 - инклинометр правого коленного сустава;
- вход 5 - инклинометр правого голеностопного сустава.

Для каждого инклинометра задается собственная градуировочная характеристика, которая используется для перевода кода АЦП в значение угла. На основе полученных значений углов производится оценка угловой скорости в суставах по методу наименьших квадратов для линейной аппроксимации. Кроме инклинометров, к ПОС подключен MEMS-сенсор Analog Devices ADIS16407BMLZ, на базе которого реализована вертикаль, позволяющая получить отклонение торса ЭРК от вектора силы тяжести в фронтальной и сагиттальной плоскостях.

Совокупность углов сгибания и угловых скоростей шести суставов ЭРК и двух углов отклонения от вертикали описывает состояние ЭРК в один момент времени. Система углов, определяющих состояние экзоскелета, приведена на рисунке 3.1.

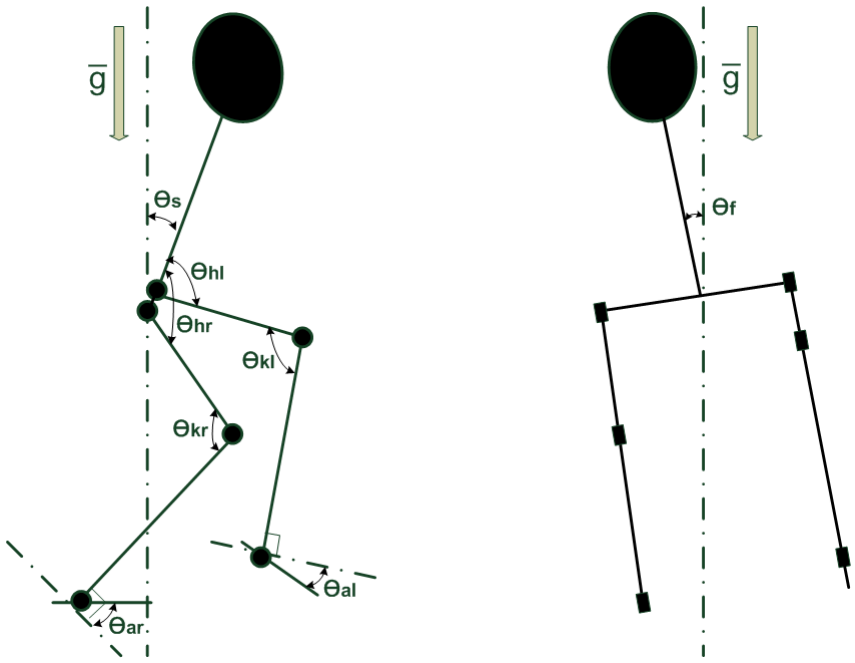

Рисунок 3.1 - Система углов, определяющих состояние ЭРК

Для получения информации о текущем состоянии ЭРК необходимо:

1. создать 6 объектов типа CalibrationTable (по одному для каждого сустава ЭРК);
2. вызвать метод CalibrationTable::LoadFromFile() для каждого объекта типа CalibrationTable. В качестве параметра указать абсолютный путь к файлам с градуировочными характеристиками сенсоров суставов:
    - home/root/LeftThighSensor.clb для левого бедра;
    - home/root/LeftKneeSensor.clb для левого колена;
    - home/root/LeftAnkleSensor.clb для левого голеностопа;
    - home/root/RightThighSensor.clb для правого бедра;
    - home/root/RightKneeSensor.clb для правого колена;
    - home/root/RightAnkleSensor.clb для правого голеностопа.
3. создать объект типа NavSensors (измеритель);
4. вызвать метод NavSensors::SetCalibrationTable() 6 раз, установив для каждого датчика свою градуировочную характеристику. В качестве параметров в метод передаются сама характеристика и индекс входа ПОС (0 - 5, отображение индексов входов на карту суставов ЭРК задается перечислением InclinometerSensorID);
5. получают интерфейс типа NavSensors посредством вызова глобальной функции GetNavSensors();
6. активируют соединение с навигационными датчиками посредством вызова у интерфейса NavSensors метода Connect() с указанием необходимой частоты обновления показаний датчиков (800 - 1000 Гц) и ширины окна медианного фильтра (3 отсчета);

7. вызвать метод NavSensors::GetData(), который возвращает структуру типа NavSensorsData. Поле NavSensorsData::angles представляет собой массив углов сгибания суставов (градусы), зафиксированных для шести входов ПОС (0 - 5, отображение индексов входов на карту суставов ЭРК задается перечислением InclinometerSensorID). Поле NavSensorsData::rotation_speeds представляет собой массив угловых скоростей (градусы в секунду), оцененных для шести суставов (0 - 5, отображение индексов входов на карту суставов ЭРК задается перечислением InclinometerSensorID). Поля NavSensorsData::sagittal_incline и NavSensorsData::frontal_incline содержат отклонения торса от вектора силы тяжести в сагиттальной и фронтальной плоскостях соответственно (градусы). Отклонения вперед и вправо положительные, назад и влево отрицательные.

Пример 3.10.1:

```
//...

//градуировочная характеристика датчика лев. бедра
CalibrationTable LeftThighSensorCalTable;

//градуировочная характеристика датчика лев. колена
CalibrationTable LeftKneeSensorCalTable;

//градуировочная характеристика датчика лев. голеностопа
CalibrationTable LeftAnkleSensorCalTable;

//градуировочная характеристика датчика прав. бедра
CalibrationTable RightThighSensorCalTable;

//градуировочная характеристика датчика прав. колена
CalibrationTable RightKneeSensorCalTable;

//градуировочная характеристика датчика прав. голеностопа
CalibrationTable RightAnkleSensorCalTable;

NavSensors * POS = GetNavSensors(); //интерфейс системы сенсоров

//Устанавливаем градуировочн. характеристики датчиков
POS->SetCalibrationTable(LeftThighSensorCalTable, LeftThighSensorID);
POS->SetCalibrationTable(LeftKneeSensorCalTable, LeftKneeSensorID);
POS->SetCalibrationTable(LeftAnkleSensorCalTable,  LeftAnkleSensorID);
POS->SetCalibrationTable(RightThighSensorCalTable, RightThighSensorID);
POS->SetCalibrationTable(RightKneeSensorCalTable, RightKneeSensorID);
POS->SetCalibrationTable(RightAnkleSensorCalTable, RightAnkleSensorID);

bool connected = POS -> Connect(200, 3);//активируем соединение с датчиками, частота опроса - 200 Гц
if(!connected) // если функция вернула false, то не удалось подключится
{
      std::cout << "Failed to establish connection with sensors" << std::endl;
      NavSensorsError error = POS -> GetLastError(); //получаем код ошибки
      std::cout << GetErrorDescription(error) << std::endl; //выводим описание ошибки
}

while(true) //цикл опроса состояния ЭРК
{
        //фиксируем текущие показания всех сенсоров, подключенных к ПОС
        NavSensorsData curr_state=POS -> GetData();

        //Вывод данных о состоянии ЭРК на консоль
        cout<<"*******************************************************"<<endl;
        cout<<"Left thigh: "<<curr_state.angles[LeftThighSensorID];
        cout<<" " << curr_state.rotation_speeds[LeftThighSensorID] <<endl;
        cout<<"Left knee: "<<curr_state.angles[LeftKneeSensorID];
        cout<<" " << curr_state.rotation_speeds[LeftThighSensorID] <<endl;
        cout<<"Left ankle: "<<curr_state.angles[LeftAnkleSensorID];
        cout<<" " << curr_state.rotation_speeds[LeftThighSensorID] <<endl;
```

```
        cout<<"Right thigh: "<<curr_state.angles[RightThighSensorID];
        cout<<" " << curr_state.rotation_speeds[LeftThighSensorID] <<endl;
        cout<<"Right knee: "<<curr_state.angles[RightKneeSensorID];
        cout<<" " << curr_state.rotation_speeds[LeftThighSensorID] <<endl;
        cout<<"Right ankle: "<<curr_state.angles[RightAnkleSensorID];
        cout<<" " << curr_state.rotation_speeds[LeftThighSensorID] <<endl;
        cout<<"Sagittal incline: "<<curr_state.sagittal_incline<<endl;
        cout<<"Frontal incline: "<<curr_state.frontal_incline<<endl;
}
//...
```

3.11 Демонстрация работоспособности электроприводов с помощью программы group-of-engines

После развертывания ПО ЭРК на микропроцессорной плате BeagleBoard XM в папке /opt/exo/bin появится программа group-of-engines. Данная программа предназначена для демонстрации работоспособности электроприводов и выполнения тестов. Программа обеспечивает независимое управление приводами с клавиатуры ПК.

Программа позволяет выбрать протокол управления контроллером привода:

- Протокол модернизированного контроллера BLSD-xx (с токовым датчиком) - NIFTY;
- Стандартный протокол управления контроллером BLSD-xx - ELECTROPRIVOD.

После запуска программа предлагает протокол управления контроллером привода, ввести идентификатор привода, выбрать количество управляемых приводов (1 или 2) и выбрать режим управления приводами или тестовый режим:

- 'w' - отправка кода токового АЦП с ограничением по углу (желаемый код вводится с клавиатуры) - протокол NIFTY или ELECTROPRIVOD;
- 's' - отправка кода токового АЦП (желаемый код вводится с клавиатуры) - протокол NIFTY или ELECTROPRIVOD;
- 'd' - отправка кода токового АЦП по синусоидальному закону (параметры вводятся с клавиатуры) - протокол NIFTY или ELECTROPRIVOD;
- 'r' - отправка кода токового АЦП по пилообразному закону (параметры вводятся с клавиатуры) - протокол NIFTY или ELECTROPRIVOD;
- 'x' - режима качения - протокол NIFTY или ELECTROPRIVOD;
- 'z' - расчет среднего времени (мс) отправки команды на контроллер - протокол NIFTY или ELECTROPRIVOD;
- 'a' - отключение питания привода - протокол NIFTY;
- 'f' - выбор режима ответа контроллера на групповую команду.

3.11.1 Режим задания кода АЦП

В данном режиме программа выполняет следующие действия:

- Запрашивает отправляемый код АЦП ("Set ADCcode:");
- После ввода оператором параметров программа устанавливает заданный код АЦП для данного привода.

3.11.2 Режим динамического изменения кода АЦП по синусоидальному закону

В данном режиме программа выполняет следующие действия:

- Запрашивает максимальное значение кода АЦП (0-250) ("Set sin amplitude:");
- Запрашивает частоту (Гц) ("Set sin frequency:");
- Запрашивает период следования отсчетов (мс) ("Set sampling period:");
- Запрашивает количество отчетов ("Set number of samples:");
- После ввода оператором параметров программа устанавливает код АЦП по синусоидальному закону.

3.11.3 Режим динамического изменения кода АЦП по линейному закону

В данном режиме программа выполняет следующие действия:

- Запрашивает максимальное значение кода АЦП (0-250) ("Set pila amplitude:");
- Запрашивает период следования отсчетов (мс) ("Set sampling period:");
- Запрашивает количество отчетов ("Set number of samples:");
- После ввода оператором параметров программа устанавливает код АЦП по линейному закону.

3.11.4 Режим качания

В режиме качания (циклическое сгибание/разгибание сустава ЭРК) программа выполняет следующие действия:

- Запрашивает код токового АЦП ("Set ADCcode: ");
- Запрашивает минимальный угол сгиба сустава, градусы ("Set min angle: ");
- Запрашивает максимальный угол сгиба сустава, градусы ("Set max angle: ");
- Запрашивает количество циклов сгибания/разгибания ("Set number of iterations: ");
- После ввода оператором параметров программа циклически сгибает/разгибает сустав ЭРК.

3.11.5 Расчет среднего времени (мс) отправки команды на контроллер

Для расчета среднего времени отправки команды на контроллер программа выполняет следующие действия:

- Запрашивает код токового АЦП ("Set ADCcode: ");
- Запрашивает количество команд для отправки("Set number of outgoing commands:");
- После ввода оператором параметров программа выдает времени отправки команды (мс).

3.11.6 Режим отключения питания привода

При выборе данного режима программа автоматически отправляет команду отключения питания привода.

3.11.7. Выбор режима ответа контроллера на групповую команду

Программа позволяет выбрать режим ответа контроллера на групповую команду (принимать ответ или нет). Для задания режима необходимо ввести:

- 0 - не принимаем ответ;
- 1 - принимаем ответ.

3.12 Получение показаний датчиков с помощью программы nav_sensors

Для тестирования и проведения измерений точности системы датчиков ЭРК было разработано приложение nav_sensors. Данная программа имеет несколько режимов работы.

3.12.1 Режим отображения отклонений оси устройства от вектора силы тяжести

Для запуска режима отображения отклонений оси устройства от вектора силы тяжести используется ключ "-n":

```
nav_sensors -n
```

3.12.2 Режим отображения показаний сенсора ADIS16407

Для запуска режима отображения показаний сенсора ADIS16407 используются ключи "-a" - для отображения показаний акселерометра, "-m" - для отображения показаний магнитометра, "-g" - для отображения показаний гироскопа.

```
nav_sensors -a
nav_sensors -m

nav_sensors -g
```

3.12.3 Режим отображения показаний инклинометров

Для запуска режима отображения показаний инклинометров используется ключ "-i [n]", где n= 0...5 - номер канала.

```
nav_sensors -i 2
```

3.12.4 Режим записи показаний датчиков в файл

Для запуска режима записи показаний датчиков в файл используется ключ "-w [t] [q]", где t - период захвата данныхв миллисекундах, q - ключ отмены вывода в консоль.

```
nav_sensors -w 100
```

Данные записываются в файл "AllParameters.csv". Файл расположен в директории исполняемого файла nav_sensors.

3.12.5 Режим продолжительных измерений вертикали

Для запуска режима продолжительных измерений вертикали используется ключ "-t [i]", где i - количество измерений. Измерения проводятся раз в минуту с момента запуска.

```
nav_sensors -t 120
```

Данные записываются в файл "LongTimeResearchResults.csv". Файл расположен в директории исполняемого файла nav_sensors.

4 Входные и выходные данные

4.1 Градуировочные характеристики инклинометров

Градуировочные характеристики инклинометров хранятся в текстовых файлах. Формат файла с градуировочной характеристикой инклинометра:

<код АЦП, беззнаковое целое 2 байта> <установленный угол, реальное двойной точности, градусы>

<код АЦП, беззнаковое целое 2 байта> <установленный угол, реальное двойной точности, градусы>

<код АЦП, беззнаковое целое 2 байта> <установленный угол, реальное двойной точности, градусы>

...

<код АЦП, беззнаковое целое 2 байта> <установленный угол, реальное двойной точности, градусы>

Путь по умолчанию к файлам с градуировочными характеристиками *home/root*:

1. LeftThighSensor.clb для левого бедра;
2. LeftKneeSensor.clb для левого колена;
3. LeftAnkleSensor.clb для левого голеностопа;
4. RightThighSensor.clb для правого бедра;
5. RightKneeSensor.clb для правого колена;
6. RightAnkleSensor.clb для правого голеностопа.

4.1 Градуировочные характеристики приводов

Градуировочные характеристики приводов хранятся в текстовых файлах. Формат файла с градуировочной характеристикой привода:

<код АЦП, беззнаковое целое 2 байта> <момент, реальное двойной точности, нм>
<код АЦП, беззнаковое целое 2 байта> <момент, реальное двойной точности, нм>
<код АЦП, беззнаковое целое 2 байта> <момент, реальное двойной точности, нм>

...

<код АЦП, беззнаковое целое 2 байта> <момент, реальное двойной точности, нм>

Путь по умолчанию к файлам с градуировочными характеристиками *home/root*:

1. LeftThighEngine.clb для левого бедра;
2. LeftKneeEngine.clb для левого колена;
3. LeftAnkleEngine.clb для левого голеностопа;
4. RightThighEngine.clb для правого бедра;
5. RightKneeEngine.clb для правого колена;
6. RightAnkleEngine.clb для правого голеностопа.

Лист регистрации изменений

Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подпись	Дата
Изм	измененных	замененных	новых	аннулированных					