EduExo Library Documentation

Introduction:

The EduExo Library is a collection of functions that can be used to control the movements of the exoskeleton.

The library is organized into two main folders: the library folder and the examples folder. The library folder contains the functions and source code for the EduExo Library. This folder is further organized into three subfolders: Communications, Sensor Readings, and Filters. Each subfolder contains functions related to a specific theme, providing an easy-to-navigate structure for users.

The examples folder contains sample control programs that demonstrate how to use the functions in the library. These examples are designed to help users get started with the EduExo Library and provide guidance on using the functions effectively.

Constants declared in this library:

```
servoAnalogPin = A3; //Servo connection analog pin

servoDigitlaPin =3; //Servo connection digital pin

Aux1Pin = A1;  //AUX connection for EMG

Aux2Pin = A2;  //Extra AUX connection

LEDPin = 4;   //LED Pin

Button1 = 9;  //Button 1 connection pin

Button2 = 8;  //Button 2 connection pin

forcePin = A4; //Force sensor analog connection pin

forceOffset = 260; //Offset value for the force sensor
```

Example Usage Code:

- ### toggle_pin:

```
#include <EduExo.h> //include EduExo library
toggle_pin(pin_number); //if the pin is currently HIGH, set it to LOW; otherwise, set it to HIGH
#include "DebounceButton.h"
```

- ### debounce_button:

```
#include <EduExo.h> //include EduExo library
void setup() {
  pinMode(buttonPin, INPUT); //setup the buttonPin as INPUT
}
void loop() {
 if (DebounceButton(buttonPin,debounceDelay))
{ // Button was pressed, do something}
}
```

- emg_sensor:

```
#include <EduExo.h> //include EduExo library
int emgValue= emgIs(aux1Pin); //read emg signal from AUX connection 1
```

- exp_filter:

```
#include <EduExo.h> //include EduExo library
ExpFilter myFilter(alpha); //declare object of type ExpFiter and with parameter alpha
void setup{
....}
void loop{
float filteredSignal = myFilter.exponentialFilter(pin); //calculate the filtered signal from pin
....}
```

- force_sensor:

```
#include <EduExo.h> //include EduExo library
int forceValue = forceIs(forcePin, forceOffset); //calculate the force value from the force sensor after calibrating it by subtracting the forceOffset.
```

- moving_average:

```
#include <EduExo.h> //include EduExo library
float sensorAverage = movingAverage(int windowSize,int sensorPin, int reading_delay); //calculate the moving average value of [windowSize] readings  and a delay of [reading_delay] ms between readings on pin [sensorPin]
```

- servo_sensor:

```
#include <EduExo.h> //include EduExo library
int servo_pos(int servoAnalogInPin) //read the servo position from servoAnalogInPin (10bit value)
int servo_pos_deg(int servoAnalogInPin, float sValue90, float sValue0) //read the servo position from servoAnalogInPin and calibrate it (in degrees). sValue0 is the value of the Angle sensor when it is at 0 degrees and sValue90 is the value of the Angle sensor when it is at 90 degrees
```

- wifi_connect:

```
#include <EduExo.h> //include EduExo library
WiFiNINA_connect wifi; //declare object of type WiFiNINA_connect
....
```

void setup() {

....

wifi.begin(ssid, user, pass); //.begin method to connect to wifi (WPA2 Enterprise) with parameters [ssid, username, password]

wifi.printCurrentNet() //print SSID, RSSI, encryptionType, local IP of the Wifi connection

}

- ## BLE_connect:

#include <EduExo.h> //include EduExo library

BLE_connect ble; //declare object of type BLE_connect

....

void_setup{

....

ble.begin(); //.begin method to initialize and setup the BLE connection

}

void loop{

ble.loop() //keep up to date with the BLE connection

....}

- ## UDP_wifi_read:

#include <EduExo.h> //include EduExo library

MyUDP myUDP; //declare object of type MyUDP

void setup{

myUDP.begin(ssid,username,password,port); //connect Wi-Fi (WPA2 Enterprise) and enable receiving data on [port]

....

}

void loop{

....

char buffer[sizeofBuffer]; //

myUPD.readPacket(buffer,sizeofBuffer); //read data as a characters array buffer of size sizeofBuffer

myUDP.sendPacket(data,sizeof(data),IPAddress(0,0,0,0),Port); //send data to the reciever's IP-Address at Port [Port]

}

- ## BLE_data:

#include <EduExo.h> //include EduExo library

BLEData BLEData(A5); //create object of type BLEData and A5 as analogPin to read data from (don't add the AnalogPin if you are not reading data from sensor)

```
void setup() {
…
while (!Serial);  //wait until the serial port is ready to receive data
 sensor.begin(); //initialize the BLE module and starts advertising the BLE service

}
void loop{
…
 sensor.sendSensorValue(); //read the sensor value of the chosen AnalogPin every 200ms and  print it on the
serial monitor
 sensor.readString(); //read the string sent from another device via BLE and  print it on the serial monitor
 sensor.sendString(“H”); //send the string “H” to another device via BLE and print it on the serial monitor

}
```