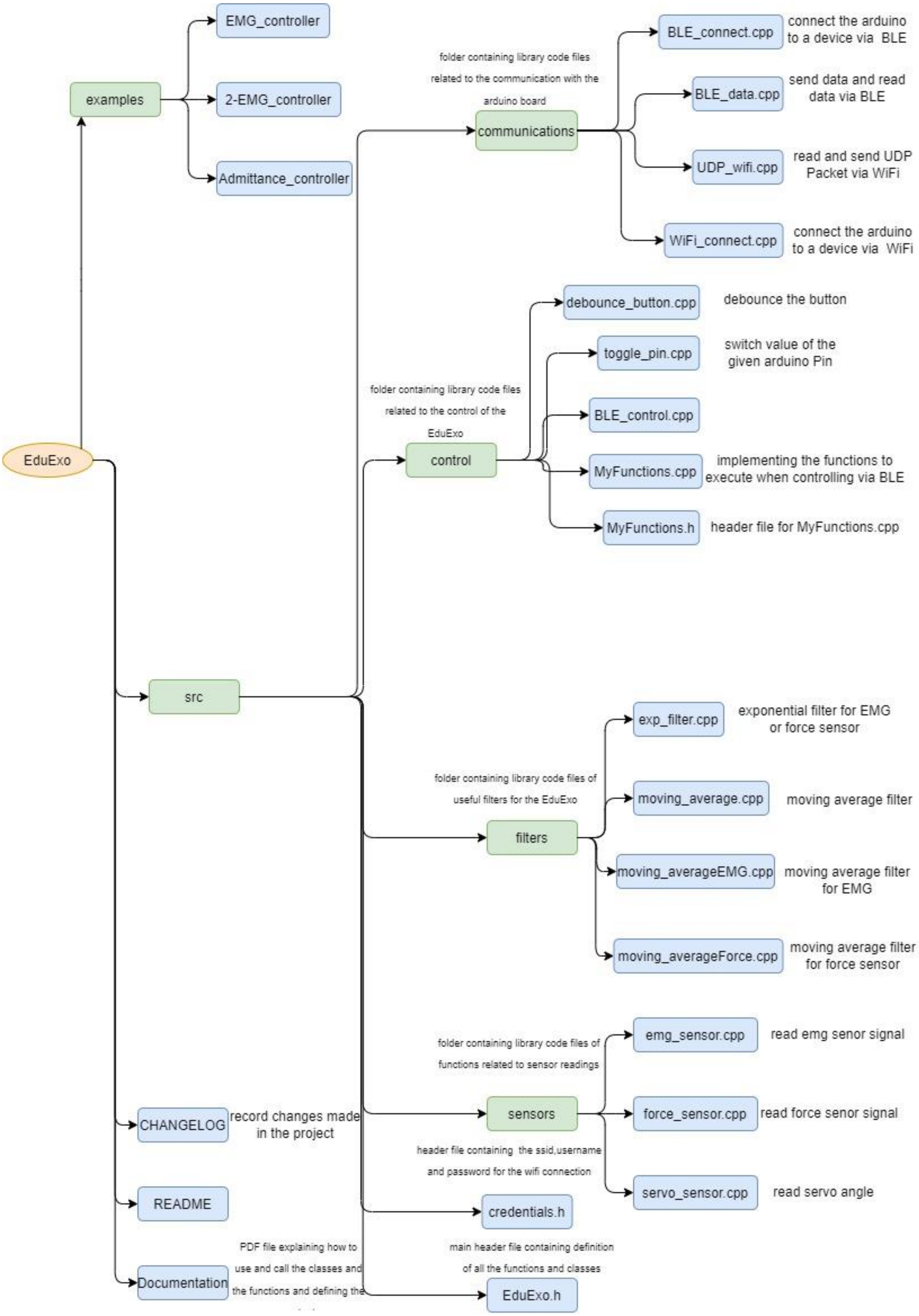
EduExo
examples
EMG_controller
2-EMG_controller
Admittance_controller
src
folder containing library code files related to the communication with the arduino board
communications
BLE_connect.cpp
connect the arduino to a device via BLE
BLE_data.cpp
send data and read data via BLE
UDP_wifi.cpp
read and send UDP Packet via WiFi
WiFi_connect.cpp
connect the arduino to a device via WiFi
folder containing library code files related to the control of the EduExo
control
debounce_button.cpp
debounce the button
toggle_pin.cpp
switch value of the given arduino Pin
BLE_control.cpp
MyFunctions.cpp
implementing the functions to execute when controlling via BLE
MyFunctions.h
header file for MyFunctions.cpp
folder containing library code files of useful filters for the EduExo
filters
exp_filter.cpp
exponential filter for EMG or force sensor
moving_average.cpp
moving average filter
moving_averageEMG.cpp
moving average filter for EMG
moving_averageForce.cpp
moving average filter for force sensor
folder containing library code files of functions related to sensor readings
sensors
emg_sensor.cpp
read emg senor signal
force_sensor.cpp
read force senor signal
servo_sensor.cpp
read servo angle
header file containing the ssid,username and password for the wifi connection
credentials.h
main header file containing definition of all the functions and classes
EduExo.h
CHANGELOG
record changes made in the project
README
Documentation
PDF file explaining how to use and call the classes and the functions and defining the

Library usage explanation:

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.
```

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=analogRead(forcePin) ; //read the force sensor value from [forcePin]
int forceCalibrated = forceIs(forceValue, forceOffset); //calculate the force value after calibrating it by subtracting the forceOffset.
```

moving_average:

```
#include <EduExo.h> //include EduExo library
float Average = 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 sensorPin
```

moving_averageEMG:

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

float EMGAverage = movingAverageEMG(int windowSize,int emgPin, int reading_delay); //calculate the moving average value of [windowSize] readings and a delay of [reading_delay] ms between readings on emgPin

- ## moving_averageForce:

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

float ForceAverage = movingAverageForce(int windowSize,int forcePin, int forceOffset,int reading_delay); //calculate the moving average value of [windowSize] readings and a delay of [reading_delay] ms between readings on forcePin after calibrating it by substracting the forceOffset.

- ## moving_averageForce:

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

float sensorAverage = movingAverageForce(int windowSize,int forcePin,int forceOffset,int reading_delay); //calculate the moving average value of [windowSize] readings of the calibrated force sensor signals and a delay of [reading_delay] ms between readings on forcePin

- ## 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(); //.begin method to connect to wifi (WPA2 Enterprise) with parameters [ssid, user, password] that are defined in “credentials.h”

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(port); // connect to wifi (WPA2 Enterprise) with parameters [ssid, user, password] that are
defined in “credentials.h” 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;  //create an object of type BLEData
void setup() {
...
while (!Serial);  //wait until the serial port is ready to receive data
 BLEData.begin(); //initialize the BLE module and starts advertising the BLE service.

}
void loop{
...
 BLEData.sendSensorValue(5); //read the sensor value of the chosen AnalogPin every 200ms and print it on
the serial monitor.
 BLEData.readString(); //read the string sent to the arduino from the device via BLE and print it on the serial
monitor.
 BLEData.readInt(); //read the integer sent to the arduino from the device via BLE and print it on the serial
monitor.
 BLEData.sendString("H"); //send the string “H” to the other device via BLE and print it on the serial monitor.
 BLEData.sendInt(133); //send the integer 133 to the other device via BLE and print it on the serial monitor
}
```

- BLE_Control :

```
#include <EduExo.h>  //include EduExo library
BLE_control BLE_control; //create an object of type BLE_control
void setup() {
…
BLE_control.begin();  //initialize the BLE module and starts advertising the BLE service

}
void loop() {
…
BLE_control.executeFunction(); //Execute the functions that are declared in the MyFunctions files
}
```