```
                    // comms.cpp
#include <stdio.h>
#include <stdlib.h>
#include <ftdi.hpp>
#include "comms.h"

Ftdi::Context ftdi ; // biblioteca C++ ftdi Ã© um objeto global

// FTDI_INIT_COMMUNICATION

int ftdi_init_communication(void)
{
 if(ftdi.open("i:0x0403:0x6001") < 0)
 {
  fprintf(stderr,"ERROR_FTDI_OPEN_PORT%s\n",ftdi.error_string()) ;
  return ERROR_FTDI_OPEN_PORT ;
 }

 if(ftdi.set_baud_rate(1000000) < 0)
 {
  fprintf(stderr,"ERROR_FTDI_SET_BAUD_RATE%s\n",ftdi.error_string());
  return ERROR_FTDI_SET_BAUD_RATE ;
 }

 printf("ftdi_open_port OK ftdi_set_baud_rate OK\n");
 return ERROR_OK ;
}
```

```
// FTDI_READ
int ftdi_read(leo2_status_package_t* package)
{
 int n = ftdi.read((unsigned char *)package , sizeof(leo2_status_package_t));

 printf("PC RECEIVING STATUS FROM BOARD\n");

 if(n != sizeof(leo2_status_package_t))
 {
  fprintf(stderr, "ERROR_FTDI_READ_STATUS read %d expected %ld: %s\n", n,
sizeof(leo2_status_package_t), ftdi.error_string()) ;
  return ERROR_FTDI_READ_STATUS ;
 }

 printf("ftdi_read OK\n");
 return ERROR_OK ;
}

// INTERPRET_STATUS
int interpret_status(leo2_status_package_t* package , leo2_status_t* status)
{
 int i ;
```

```
unsigned char byte_checksum = 0 ;

printf("init INTERPRET STATUS\n");

if(package->header_1 != 0xFF || package->header_2 != 0xFF)
{
 printf("ERROR_HEADER\n") ;
 return ERROR_HEADER ;
}
printf("STATUS_PACKAGE_HEADER_OK\n");

printf("Conferindo Checksum do pacotede estados\n");
for(i = 0 ; i < sizeof(leo2_status_package_t) - 1 ; i++)
{
 byte_checksum += ((unsigned char *)package)[i] ;
 printf("Status_package byte %d %d\n",i, ((unsigned char *)package)[i]);
}

byte_checksum = 255 - byte_checksum;

printf("byte_checksum calculado %d\n",byte_checksum);

printf("status package byte checksum %d\n",package->checksum);

if(byte_checksum != package->checksum)
{
 printf("ERROR_CHECKSUM\n") ;
 return ERROR_CHECKSUM ;
}
printf("STATUS_PACKAGE_CHECKSUM OK\n");
```

```
status->status = package->status ;

printf("STATUS %d \n",package->status);

for(i = 0 ; i < NUM_ENC ; i++)
{
 unsigned int word_pos = network_to_word(&package->pos[2*i]);
 unsigned int word_speed = network_to_word(&package->speed[2*i]) ;

 status->pos[i] = word_pos * 0.088 ;

 printf("Position Raw -> Degrees -- %d -> %f\n", word_pos,status->pos[i]) ;

 // WHELL MODE

 if(word_speed < 1024)
 {
  printf("CW direction\n");
  status->speed[i] = 0.117 * word_speed; // CW
 }
 else
 {
  printf("CCW direction speed\n");
  status->speed[i] = (word_speed - 1024) * -0.117 ; // CCW
 }
 printf("Speed Raw -> radPerSec %d -> %f\n",word_speed,status->speed[i]);
}

          //for(i = 0 ; i < 2 ; i++)
            //{
```

```
            //unsigned int word_voltage = unsigned int network_to_word(&package->voltage[i]);
            //status->voltage[i] = word_voltage * 0.0009775 ;
            //printf"Voltage Raw -> Voltage volts %d -> %f\n",word_voltage,status->voltage[i]);
           //}



 printf("interpret_status ERROR_OK\n");
 printf("end INTERPRET STATUS\n");

 return ERROR_OK ;
}



// PREPARE PACKAGE
void prepare_package(leo2_command_t* command , leo2_command_package_t* package)
{
 int i ;

 unsigned char byte_checksum = 0 ;

 printf("init PREPARE_PACKAGE\n");

 package->header_1 = 0xFF ;
```

```
 package->header_2 = 0xFF ;


 printf("Command Package header_1 %d\n",package->header_1);

 printf("Command Package header_2 %d\n",package->header_2);


 package->length = 2 * NUM_ACT ;

 printf("Command Package length %d \n",package->length);


 package->command = command->command ; // SerÃ¡ preenchido pela funcao que  preenche
o controlador e leo2command

 printf("Command package command %d\n",package->command);


 switch (package->command){


 case CMD_POSITION :

 for(i = 0 ; i < NUM_ACT ; i++)

 {

  unsigned int word = command->data[i] / 0.088 ;


  if(word > 4096)

  word = 4096;

  if(word < 0)

  word = 0;


  printf("Command Package Data Raw Word %d Command Rad per sec %f\n",word,command-
>data[i]);


  word_to_network(word, &package->data[2*i]);

  printf("Command package first byte %d\n",package->data[0+2*i]);

  printf("Command package second byte %d\n",package->data[1+2*i]);

 }

 break ;
```

```
case CMD_SPEED :
for(i = 0 ; i < NUM_ACT ; i++)
{
 //int direction ;
 int word = command->data[i]/0.117;
 printf("Command Data rad per sec %f\n",command->data[i]);

 if(word > 0)
 {
   if (word > 1023) word = 1023;

  printf("CCW command\n");
 }
 else
 {
  if (word < -1023) word = -1023;

  word = 1024 - word;
  printf("CW command\n");
 }

printf("Command package Word DATA RAW %d\n",word);

word_to_network(word,&package->data[2*i]);

// CONSERTAR!!!!!!!

printf("package data first byte %d\n",package->data[0+2*i]); // ---- HAVE TO CHECK

printf("package data second byte %d\n",package->data[1+2*i]); // ---- HAVE TO CHECK
```

```
}
break ;

case CMD_TORQUE : // consertar
for(i = 0 ; i < NUM_ACT ; i++)
{
 int torque = command->data[i]/0.0009775;

 printf("Command Data Word Raw %d Command Package SI unit %f\n",torque, command->data[i]);

 if (torque < 0)
 {
  printf("CCW direction torque\n");
  if(torque < -1023)
  torque = -1023;
  torque = -1*torque; // 0-1023  CCW TORQUE
 }
 else
 {
 printf("CW direction torque\n");
 if(torque > 0 && torque > 1023)
 torque = 1023;

 torque = 1024 + torque; // 1024-2048 - CW TORQUE
 }

 word_to_network(torque,&package->data[2*i]);

 printf("Command package first byte %d\n",package->data[0+2*i]);
 printf("Command package second byte %d\n",package->data[1+2*i]);
```

```
 }
 break ;
 } // end switch

 printf("Checksum Calculus\n");
 printf("show Command Package byte to byte\n");
 for(i = 0 ; i < sizeof(leo2_command_package_t) - 1 ; i++)
 {
 byte_checksum += ((unsigned char *)package)[i] ;
 printf("%d\n",((unsigned char *)package)[i]);
 }
 printf("checksum calculated %d\n",255 - byte_checksum);

 package->checksum = 255 - byte_checksum ;
 printf("Command package checksum %d\n",package->checksum);

 if(255-byte_checksum != package->checksum)
 printf("ERROR COMMAND PACKAGE CHECKSUM\n");
 else
 printf("COMMAND PACKAGE CHECKSUM OK\n");

 printf("end PREPARE PACKAGE\n");
}



// FTDI_WRITE_PACKAGE
void ftdi_write_package(leo2_command_package_t* package)
{
 ftdi.write((unsigned char *)package , sizeof(leo2_command_package_t));
 printf("pc write command package to board\n");
```

```
}
```

```
// FTDI_CLOSE_COMMUNICATION
void ftdi_close_communication(void)
{
 ftdi.close() ;
}
```