```
/*
    Ivan Abreu Studio.

    09/07/2018

    This code controls a set of 3 stepper motors, 2 of them are
    used to turn the head of the system host, the last one to
    turn the needle of a big compass.

    The target of this pieces is force the host to always see
    to south and express it with a big model of a compass.

    Changelog:
    V0.1 First development
    V0.2.1 Test sequence for L motor
    V0.2.2 Test for 3 motors single direction
    V0.3 Merge Absolute orientation sensor code
    V0.4 ShortestWay Added
    V0.5 Continuous Movement Added
    V0.6 Sensor compass calibration added
    V0.6.1 Compass Servo sequence fixed
    V0.6.2 Change direction sequence Added
    V0.6.3 Pull and push mirror movement in reducer steppers added
    V0.6.4 Bluetooth communication
    V0.7 Bluetooth Control

    Team
    Iván Abreu Ochoa
    Malitzin Cortes
    Beto Olguin
    Hugo Vargas

*/

//Libraries
#include <Wire.h>
#include   <Adafruit_Sensor.h>
#include   <Adafruit_BNO055.h>
#include   <utility/imumaths.h>

//Constants
const byte DIR_1 = 4;
const byte STEP_1 = 5;
const byte DIR_2 = 6;
const byte STEP_2 = 7;
const byte COMPASS_PIN [] = {8, 9, 10, 11};
const byte LEFT_MOTOR = 0;
const byte RIGHT_MOTOR = 1;
const byte COMPASS_MOTOR = 2;
const bool LEFT_DIR = 0;
const bool RIGHT_DIR = 1;
const long TIME_TESTEPS = 1000;
const bool ON = 1;
const bool OFF = 0;
const long TEST_STEPS = 999500;
const int IBWTT = 250;//In Between Wait Test Time
const  long  WORK_TIME_STEP_COMPASS  =  3000;  //uSeconds
const  long  WORK_TIME_STEP  =  900;  //uSeconds
const  byte  BASE_TH  =  15;  //Degrees
const  int  PIN_S1  =  A2;
const  int  PIN_S2  =  A3;
const  int  DETECT_S1  =  700;
const  int  DETECT_S2  =  530;
const  int  HALL_DEBOUNCE  =  100;
const  int  RING_LENGTH  =  16;
const  long  SENSOR_SAMPLE_TIME  =  80000;
const  long  TRASCIENT_TIME_UP  =  1500000;
const  long  TRASCIENT_TIME_DOWN  =  600000;

#define   BNO055_SAMPLERATE_DELAY_MS   (100)

//Objects
Adafruit_BNO055  bno  =  Adafruit_BNO055();

//Variables
bool dirMotor [] = {0, 0, 0};
long  stepMotorTime  []  =  {TIME_TESTEPS,  TIME_TESTEPS,  TIME_TESTEPS};
bool runMotor [] = {0, 0, 0};
long stepTimeTarget [] = {0, 0, 0};
long  timeNow;
bool levelMotor [] = {0, 0, 0};
long  testSteps;
byte  compassSequence  =  0;

long   AOSensorTime;
int  heading;
int  diffference;
bool   compassDirection;
int  degreesLeft;
long   workingCompassTimeStep,   workingMotorTimeStep;
long   workingDirLeft,  workingDirRight;
byte  threshold  =  BASE_TH;
bool   closeEnoughCompass,   closeEnoughLeft,   closeEnoughRight;
int    calibrationCounter;
int  lecture1;
long stepRegistry [] = {0,  0,  0};
long    lastCalibrationCounter;
long    latestCalibrationCounter;
byte  i_sensorRing  =  0;
byte  sensorRing  [RING_LENGTH];
long  sensorTime;
byte  markOne;
byte  pointOne,  pointTwo;
long  isTrascient;
int  lastResponse;
bool handShake  =  0;
String  rValueBT;
int  buffBT;
int  buffMag;


void  setup()  {
   //  Serial  Monitor  communication
   Serial.begin  (2000000);
   Serial.println  ("Setting  up");

   setPinModes  ();
   setInitialConditions  ();

   testSequence  ();

   //Wait  for  a  recongizable  number,  only  initialize  when  the
   //expected  number  is  received
   waitHandShake  ();

   setWorkingConditions  ();

   printMenu  ();
}

void  loop()  {

   /*re  position  this  in  an  specific  submenu}
      readAbsoluteOrientationSensor  ();
     shortestWayToSouth  ();
     motorDirective  ();
     runAll  ();

   */

   //Bluetooth  configuration  service.  A  menu  accesable  theough  serial  BT
   //that  determines  on-off  functions  mainly.  Also  let  you  choose  a
   //manual  calibration,  test  sequence  and  independently  confivurations
   //of  working  parameters...

   readBT  ();
   buffBT  =  rValueBT.toInt  ();
   switch  (buffBT)  {
     case  0:
       printMenu  ();
       clean  ();
       break;
     case  1:
       testSequence  ();
       clean  ();
        Serial2.println  ("Test  Sequence  Done");
       break;
     case  2:
        beginOrientationSensor  ();
       clean  ();
         Serial2.println  ("Absolute  Orientation  Sensor  started");
       break;
     case  3:
        runUntilCalibrate  ();
       clean  ();
        Serial2.println  ("Sensor  Calibrated");
       break;
     case  4:
        calibrateCompassDisc ();
       clean ();
        Serial.println ("Calibration finished");
       break;
     case 5:
       searchSouth ();
       clean ();
        Serial.println (";)");
       break;
     case 6:
       tense ();
       clean ();
       break;
     case 7:
       loose ();
       clean ();
       break;
     default:
        Serial2.println ("Try Again");
       printMenu ();
       break;
   }
}
```