|  |
| --- |
| /\*  Programa creado sobre MinIMU-9-Arduino-AHRS.  MRS.  MinIMU-9-Arduino-AHRS  Pololu MinIMU-9 + Arduino AHRS (Attitude and Heading Reference System)  Copyright (c) 2011 Pololu Corporation.  http://www.pololu.com/  MinIMU-9-Arduino-AHRS is based on sf9domahrs by Doug Weibel and Jose Julio:  http://code.google.com/p/sf9domahrs/  sf9domahrs is based on ArduIMU v1.5 by Jordi Munoz and William Premerlani, Jose  Julio and Doug Weibel:  http://code.google.com/p/ardu-imu/  \*/  // Uncomment the below line to use this axis definition:  // X axis pointing forward  // Y axis pointing to the right  // and Z axis pointing down.  // Positive pitch : nose up  // Positive roll : right wing down  // Positive yaw : clockwise  int SENSOR\_SIGN**[**9**]** **=** **{** 1**,** 1**,** 1**,** **-**1**,** **-**1**,** **-**1**,** 1**,** 1**,** 1 **};** //Correct directions x,y,z - gyro, accelerometer, magnetometer  int SENSOR\_SIGNd**[**9**]** **=** **{** 1**,** 1**,** 1**,** **-**1**,** **-**1**,** **-**1**,** 1**,** 1**,** 1 **};** //Correct directions x,y,z - gyro, accelerometer, magnetometer  // Uncomment the below line to use this axis definition:  // X axis pointing forward  // Y axis pointing to the left  // and Z axis pointing up.  // Positive pitch : nose down  // Positive roll : right wing down  // Positive yaw : counterclockwise  //int SENSOR\_SIGN[9] = {1,-1,-1,-1,1,1,1,-1,-1}; //Correct directions x,y,z - gyro, //accelerometer, magnetometer  //int SENSOR\_SIGN[9] = {1,-1,-1,-1,1,1,1,-1,-1}; //Correct directions x,y,z - gyro, //accelerometer, magnetometer  //int SENSOR\_SIGNd[9] = {1,-1,-1,-1,1,1,1,-1,-1}; //Correct directions x,y,z - gyro, //accelerometer, magnetometer  // tested with Arduino Uno with ATmega328 and Arduino Duemilanove with ATMega168  #include <Wire.h>  #include <LiquidCrystal.h> // F Malpartida's NewLiquidCrystal library  #define caraZowi 0 // =1: Operador ve CARA de Zowi. =0: Operador percibe CULO.  // LSM303 accelerometer: 8 g sensitivity  // 3.9 mg/digit; 1 g = 256  #define GRAVITY 256 //this equivalent to 1G in the raw data coming from the accelerometer  #define ToRad(x) ((x)\*0.01745329252) // \*pi/180  #define ToDeg(x) ((x)\*57.2957795131) // \*180/pi  // L3G4200D gyro: 2000 dps full scale  // 70 mdps/digit; 1 dps = 0.07  #define Gyro\_Gain\_X 0.07 //X axis Gyro gain  #define Gyro\_Gain\_Y 0.07 //Y axis Gyro gain  #define Gyro\_Gain\_Z 0.07 //Z axis Gyro gain  #define Gyro\_Scaled\_X(x) ((x)\*ToRad(Gyro\_Gain\_X))  //Return the scaled ADC raw data of the gyro in radians for second  #define Gyro\_Scaled\_Y(x) ((x)\*ToRad(Gyro\_Gain\_Y))  //Return the scaled ADC raw data of the gyro in radians for second  #define Gyro\_Scaled\_Z(x) ((x)\*ToRad(Gyro\_Gain\_Z))  //Return the scaled ADC raw data of the gyro in radians for second  #define Gyro\_Scaled\_Xd(x) ((x)\*ToRad(Gyro\_Gain\_X))  //Return the scaled ADC raw data of the gyro in radians for second  #define Gyro\_Scaled\_Yd(x) ((x)\*ToRad(Gyro\_Gain\_Y))  //Return the scaled ADC raw data of the gyro in radians for second  #define Gyro\_Scaled\_Zd(x) ((x)\*ToRad(Gyro\_Gain\_Z))  //Return the scaled ADC raw data of the gyro in radians for second  // LSM303 magnetometer calibration constants; use the Calibrate example from  // the Pololu LSM303 library to find the right values for your board  #define M\_X\_MIN -2666  #define M\_Y\_MIN -3446  #define M\_Z\_MIN -4782  #define M\_X\_MAX 4625  #define M\_Y\_MAX 3695  #define M\_Z\_MAX 4383  #define M\_X\_MINd -3212  #define M\_Y\_MINd -3273  #define M\_Z\_MINd -2653  #define M\_X\_MAXd 3505  #define M\_Y\_MAXd 3260  #define M\_Z\_MAXd 4601  #define Kp\_ROLLPITCH 0.02  #define Ki\_ROLLPITCH 0.00002  #define Kp\_YAW 1.2  #define Ki\_YAW 0.00002  // Mesa botonera: \*\*\* ELIMINAR LED Y BUZZER  #define pin\_pulsador1 9  #define pin\_interruptor 11  //#define pin\_led 7 //OLD: 12. 7 para debug en cambio a versión raspberry.  //#define pin\_buzz 8  #define umbralError 0.8  #define ErrorArtMAX 1  /\*For debugging purposes\*/  //OUTPUTMODE=1 will print the corrected data,  //OUTPUTMODE=0 will print uncorrected data of the gyros (with drift)  #define OUTPUTMODE 1  //#define PRINT\_DCM 0 //Will print the whole direction cosine matrix  #define PRINT\_ANALOGS 0 //Will print the analog raw data  #define PRINT\_EULER 1 //Will print the Euler angles Roll, Pitch and Yaw  #define STATUS\_LED 13  //Pines pinales de carrera zapateros  #define FCV1 2  #define FCH1 3  #define FCV2 4  #define FCH2 5  //pines leds  #define Led\_rojo 14  #define Led\_verde 16  #define Led\_azul 15  #define Led\_reserva 17  #define Pin\_buzz 8  #define round(a) (int)(a + 0.5) //Para redondear  float G\_Dt **=** 0.02**;** // Integration time (DCM algorithm)  // We will run the integration loop at 50Hz if possible  long timer **=** 0**;** //general purpuse timer  long timer\_old**;**  long timer\_restart **=** 0**;**  long timer24 **=** 0**;** //Second timer used to print values  int AN**[**6**];** //array that stores the gyro and accelerometer data  int AN\_OFFSET**[**6**]** **=** **{** 0**,** 0**,** 0**,** 0**,** 0**,** 0 **};**  //Array that stores the Offset of the sensors  int ANd**[**6**];** //array that stores the gyro and accelerometer data  int AN\_OFFSETd**[**6**]** **=** **{** 0**,** 0**,** 0**,** 0**,** 0**,** 0 **};**  //Array that stores the Offset of the sensors  int gyro\_x**;**  int gyro\_y**;**  int gyro\_z**;**  int accel\_x**;**  int accel\_y**;**  int accel\_z**;**  int magnetom\_x**;**  int magnetom\_y**;**  int magnetom\_z**;**  float c\_magnetom\_x**;**  float c\_magnetom\_y**;**  float c\_magnetom\_z**;**  float MAG\_Heading**;**  int gyro\_xd**;**  int gyro\_yd**;**  int gyro\_zd**;**  int accel\_xd**;**  int accel\_yd**;**  int accel\_zd**;**  int magnetom\_xd**;**  int magnetom\_yd**;**  int magnetom\_zd**;**  float c\_magnetom\_xd**;**  float c\_magnetom\_yd**;**  float c\_magnetom\_zd**;**  float MAG\_Headingd**;**  float Accel\_Vector**[**3**]** **=** **{** 0**,** 0**,** 0 **};** //Store the acceleration in a vector  float Gyro\_Vector**[**3**]** **=** **{** 0**,** 0**,** 0 **};** //Store the gyros turn rate in a vector  float Omega\_Vector**[**3**]** **=** **{** 0**,** 0**,** 0 **};** //Corrected Gyro\_Vector data  float Omega\_P**[**3**]** **=** **{** 0**,** 0**,** 0 **};** //Omega Proportional correction  float Omega\_I**[**3**]** **=** **{** 0**,** 0**,** 0 **};** //Omega Integrator  float Omega**[**3**]** **=** **{** 0**,** 0**,** 0 **};**  float Accel\_Vectord**[**3**]** **=** **{** 0**,** 0**,** 0 **};** //Store the acceleration in a vector  float Gyro\_Vectord**[**3**]** **=** **{** 0**,** 0**,** 0 **};** //Store the gyros turn rate in a vector  float Omega\_Vectord**[**3**]** **=** **{** 0**,** 0**,** 0 **};** //Corrected Gyro\_Vector data  float Omega\_Pd**[**3**]** **=** **{** 0**,** 0**,** 0 **};** //Omega Proportional correction  float Omega\_Id**[**3**]** **=** **{** 0**,** 0**,** 0 **};** //Omega Integrator  float Omegad**[**3**]** **=** **{** 0**,** 0**,** 0 **};**  // Euler angles pierna izquierda  float roll**;**  float pitch**;**  float yaw**;**  // Euler angles pierna derecha  float rolld**;**  float pitchd**;**  float yawd**;**  float errorRollPitch**[**3**]** **=** **{** 0**,** 0**,** 0 **};**  float errorYaw**[**3**]** **=** **{** 0**,** 0**,** 0 **};**  float errorRollPitchd**[**3**]** **=** **{** 0**,** 0**,** 0 **};**  float errorYawd**[**3**]** **=** **{** 0**,** 0**,** 0 **};**  unsigned int counter **=** 0**;**  byte gyro\_sat **=** 0**;**  byte gyro\_satd **=** 0**;**  float DCM\_Matrix**[**3**][**3**]** **=** **{**  **{** 1**,** 0**,** 0 **},** **{** 0**,** 1**,** 0 **},** **{** 0**,** 0**,** 1 **}**  **};**  float DCM\_Matrixd**[**3**][**3**]** **=** **{**  **{** 1**,** 0**,** 0 **},** **{** 0**,** 1**,** 0 **},** **{** 0**,** 0**,** 1 **}**  **};**  float Update\_Matrix**[**3**][**3**]** **=** **{** **{** 0**,** 1**,** 2 **},** **{** 3**,** 4**,** 5 **},** **{** 6**,** 7**,** 8 **}** **};** //Gyros here  float Update\_Matrixd**[**3**][**3**]** **=** **{** **{** 0**,** 1**,** 2 **},** **{** 3**,** 4**,** 5 **},** **{** 6**,** 7**,** 8 **}** **};** //Gyros here  float Temporary\_Matrix**[**3**][**3**]** **=** **{**  **{** 0**,** 0**,** 0 **},** **{** 0**,** 0**,** 0 **},** **{** 0**,** 0**,** 0 **}**  **};**  float Temporary\_Matrixd**[**3**][**3**]** **=** **{**  **{** 0**,** 0**,** 0 **},** **{** 0**,** 0**,** 0 **},** **{** 0**,** 0**,** 0 **}**  **};**  // !! botonera  int estado **=** 0**;**  long timerLed **=** 0**;**  int freqLed **=** 1**;**  int pulsador\_old **=** 0**;**  int pulsador\_apagar **=** 1**;**  int num\_vueltas **=** 0**;**  double comprPitch\_izq **=** 0**;**  double comprRoll\_izq **=** 0**;**  //double comprYaw\_izq=0;  double comprPitch\_der **=** 0**;**  double comprRoll\_der **=** 0**;**  //double comprYaw\_der=0;  double contadorEstabilizacion **=** 0**;**  int flagBoton **=** 0**;**  double nuevo90\_der **=** 0**;**  double nuevo90\_izq **=** 0**;**  double nuevo0\_der **=** 0**;**  double nuevo0\_izq **=** 0**;**  double err\_pie\_izq **=** 0**;**  double err\_cadera\_izq **=** 0**;**  double err\_pie\_der **=** 0**;**  double err\_cadera\_der **=** 0**;**  double pos\_old\_pie\_izq **=** 0**;**  double pos\_old\_cadera\_izq **=** 0**;**  double pos\_old\_pie\_der **=** 0**;**  double pos\_old\_cadera\_der **=** 0**;**  int cadera\_izq\_OK **=** 0**;**  int cadera\_der\_OK **=** 0**;**  int pie\_izq\_OK **=** 0**;**  int pie\_der\_OK **=** 0**;**  int primeraVez **=** 0**;**  int numeroData **=** 0**;**  int flag\_espera\_zum **=** 0**;**  double timer\_Zum **=** 0**;**  double timer\_WD**;**  char lectura\_RB **=** 0**;**  int FCH\_1 **=** 0**;**  int FCV\_1 **=** 0**;**  int FCH\_2 **=** 0**;**  int FCV\_2 **=** 0**;**  //Comunicaciones  int slave\_add **=** 1**;** //Direccion I2C del esclavo (Zowi)  char resultado **=** '0'**;**  int result **=** 0**;**  LiquidCrystal lcd**(**0**);**  char data**[**4**]** **=** **{** 0**,** 0**,** 0**,** 0 **};**  char dataTRICK**[**4**]** **=** **{** 0**,** 0**,** 0**,** 0 **};**  double dataError**[**4**]** **=** **{** 0**,** 0**,** 0**,** 0 **};**  char trama**[**30**]** **=** **{** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,**  0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0 **};**  char tramaError**[**40**]** **=** **{** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,**  0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,**  0**,** 0**,** 0**,** 0**,** 0**,** 0 **};**  String str**;**  char temp**[**2**];**  int dataINT**[**4**]** **=** **{** 0**,** 0**,** 0**,** 0 **};**  //Comandos I2C  char C0**[**7**]** **=** "$IZUM:"**;** //Comando arrancar y cargar programa calibración ZUM  char C1**[**7**]** **=** "$ROFC:"**;** //Comando leer offset en zowi  char C2**[**7**]** **=** "$WOFC:"**;** //Comando escribir offset calibracion en zowi  char C3**[**7**]** **=** "$M90C:"**;** //Comando mover 4 servos a 90  char C4**[**7**]** **=** "$MHOC:"**;** //Comando mover 4 servos a HOME (nuevo90 calibrado)  char C5**[**7**]** **=** "$MSSC:"**;** //Comando mover 4 servos a posiciones indicadas  char C6**[**7**]** **=** "$MSxC:"**;** //Comando mover servo indicado a posicion indicada  char C7**[**7**]** **=** "$WERC:"**;** //Comando enviar error a RB  char C8**[**7**]** **=** "$FZUM:"**;** //Comando programar test en ZUM  char C9**[**7**]** **=** "$ROFF:"**;** //Comando apagar RB  char C10**[**7**]** **=** "$WSQL:"**;** //Orden de escribir a MySQL  int dato **=** 0**;**  void setup**()**  **{**  lcd**.**begin**(**16**,** 2**);**  pinMode**(**pin\_interruptor**,** INPUT\_PULLUP**);** //interruptor mesa botonera  pinMode**(**pin\_pulsador1**,** INPUT\_PULLUP**);** //pulsador botonera  // pinMode(pin\_led, OUTPUT); //led mesa botonera  pinMode**(**Pin\_buzz**,** OUTPUT**);** //buzzer  pinMode**(**Led\_rojo**,** OUTPUT**);** //leds  pinMode**(**Led\_azul**,** OUTPUT**);** //leds  pinMode**(**Led\_verde**,** OUTPUT**);** //leds  pinMode**(**Led\_reserva**,** OUTPUT**);** //leds  pinMode**(**FCH1**,** INPUT\_PULLUP**);** //Final carrera horizontal 1  pinMode**(**FCV1**,** INPUT\_PULLUP**);** //Final carrera vertical 1  pinMode**(**FCH2**,** INPUT\_PULLUP**);** //Final carrera horizontal 2  pinMode**(**FCV2**,** INPUT\_PULLUP**);** //Final carrera vertical 2  Serial**.**begin**(**115200**);**  Serial1**.**begin**(**9600**);**  pinMode**(**STATUS\_LED**,** OUTPUT**);** // Status LED  I2C\_Init**();**  Serial1**.**println**(**"Banco calibracion Zowi"**);**  lcd**.**clear**();**  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" System "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" start-up... "**);**  digitalWrite**(**Led\_rojo**,** LOW**);**  digitalWrite**(**Led\_azul**,** LOW**);**  digitalWrite**(**Led\_verde**,** LOW**);**  digitalWrite**(**Led\_reserva**,** LOW**);**  digitalWrite**(**Pin\_buzz**,** LOW**);**  digitalWrite**(**STATUS\_LED**,** LOW**);**  delay**(**1500**);**  Accel\_Init**();**  Serial1**.**println**(**"acel init done"**);**  Compass\_Init**();**  Serial1**.**println**(**"compass initdone"**);**  Gyro\_Init**();**  Serial1**.**println**(**"gyro init done"**);**  delay**(**20**);**  **for** **(**int i **=** 0**;** i **<** 32**;** i**++)** // We take some readings...  **{**  Read\_Gyro**();**  Read\_Accel**();**  **for** **(**int y **=** 0**;** y **<** 6**;** y**++)** // Cumulate values  **{**  AN\_OFFSET**[**y**]** **+=** AN**[**y**];**  AN\_OFFSETd**[**y**]** **+=** ANd**[**y**];**  **}**  delay**(**20**);**  **}**  **for** **(**int y **=** 0**;** y **<** 6**;** y**++)** **{**  AN\_OFFSET**[**y**]** **=** AN\_OFFSET**[**y**]** **/** 32**;**  AN\_OFFSETd**[**y**]** **=** AN\_OFFSETd**[**y**]** **/** 32**;**  **}**  AN\_OFFSET**[**5**]** **-=** GRAVITY **\*** SENSOR\_SIGN**[**5**];**  AN\_OFFSETd**[**5**]** **-=** GRAVITY **\*** SENSOR\_SIGNd**[**5**];**  delay**(**2000**);**  digitalWrite**(**STATUS\_LED**,** HIGH**);**  timer **=** millis**();**  delay**(**20**);**  counter **=** 0**;**  estado **=** 97**;**  primeraVez **=** 1**;**  **}**  void loop**()** //Main Loop (Maquina estados)  **{**  **if** **(**digitalRead**(**pin\_interruptor**)** **==** LOW **&&** estado **!=** 97 **&&** estado **!=** 98  **&&** estado **!=** 96**)** **{**  estado **=** 0**;**  delay**(**10**);**  **}**  **switch** **(**estado**)** **{**  **case** 96**:** //Apagado de RPI  WriteComm**(**9**);**  lcd**.**clear**();**  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" System "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**"Shutting down..."**);**  Serial1**.**println**(**"Apagando"**);**  delay**(**15000**);**  lcd**.**clear**();**  lcd**.**noDisplay**();**  digitalWrite**(**Led\_rojo**,** LOW**);**  parpadeoLed**(**0**);** //desactivamos parpadeo led azul  digitalWrite**(**Led\_azul**,** LOW**);**  digitalWrite**(**Led\_verde**,** LOW**);**  digitalWrite**(**Led\_reserva**,** LOW**);**  digitalWrite**(**Pin\_buzz**,** LOW**);**  digitalWrite**(**STATUS\_LED**,** LOW**);**  **break;**  **case** 97**:** //Arranque de RPI //esperamos que la RPI envíe un 1 a la Mega  parpadeoLed**(**1**);**  **if** **(**Serial**.**available**()** **>=** 1**)** **{**  lectura\_RB **=** Serial**.**read**();**  Serial1**.**println**(**"lectura serial RB:"**);**  Serial1**.**println**(**lectura\_RB**);**  **}**  //Dependiendo del estado del switch vamos a un estado u otro:  **if** **((**lectura\_RB **==** '1'**)** **&&** **(**digitalRead**(**pin\_interruptor**)** **==** HIGH**))** **{**  estado **=** 14**;**  sonidostart**();**  parpadeoLed**(**0**);**  **}**  **else** **if** **((**lectura\_RB **==** '1'**)** **&&** **(**digitalRead**(**pin\_interruptor**))** **==** LOW**)** **{**  estado **=** 0**;**  sonidostart**();**  parpadeoLed**(**0**);**  **}**  **break;**  //Llegamos a estado 0 al subir el switch desde cualquier estado  **case** 0**:**  digitalWrite**(**Led\_rojo**,** LOW**);**  parpadeoLed**(**0**);** //desactivamos parpadeo led azul  digitalWrite**(**Led\_azul**,** LOW**);**  digitalWrite**(**Led\_verde**,** LOW**);**  digitalWrite**(**Led\_reserva**,** LOW**);**  digitalWrite**(**Pin\_buzz**,** LOW**);**  digitalWrite**(**STATUS\_LED**,** LOW**);**  **if** **(**digitalRead**(**pin\_pulsador1**)** **==** 1 **&&** pulsador\_apagar **==** 1**)** **{**  //Apagado del sistema:  delay**(**5**);**  timer\_restart **=** millis**();**  pulsador\_apagar **=** 0**;**  **}**  **else** **if** **(**digitalRead**(**pin\_pulsador1**)** **==** 0**)** **{**  timer\_restart **=** millis**();**  pulsador\_apagar **=** 1**;**  **}**  **if** **((**millis**()** **-** timer\_restart**)** **>=** 2000 **&&** digitalRead**(**pin\_pulsador1**)** **==** 1**)** **{**  //si mantenemos el pulsador 2 segundos apagamos sistema  estado **=** 96**;** //estado para apagar  **break;**  **}**  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Set to "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" CALIBRATE pos. "**);**  ////Serial1.println("Estado 0");  /\* interruptor en posición 0. Desde cualquier estado podremos volver  al estado 0 poniendolo a 0. Los servos energizados reciben el valor 90,  el operario deberá montar y atornillar los piés del ZOWI en la posición  más correcta posible.  Se deben colocar los zapatos en la plataforma ntes de bajar el interruptor\*/  //al cambiar el interruptor pasamos al funcionamiento del sistema  **if** **(**digitalRead**(**pin\_interruptor**)** **==** HIGH**)** **{**  estado **=** 14**;**  delay**(**10**);**  primeraVez **=** 1**;**  **};**  **break;**  **case** 14**:** //calibracion horizontal tras pulsar el boton  Serial1**.**println**(**"Estado 14: Espera colocar zapatos y pulsar boton"**);**  //deben estar los zapatos insertados para calibrar las imus  FCH\_1 **=** digitalRead**(**FCH1**);**  FCH\_2 **=** digitalRead**(**FCH2**);**  **if** **(**FCH\_1 **==** HIGH **||** FCH\_2 **==** HIGH**)** **{**  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Place shoes in "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" Horizontal box "**);**  **}**  **if** **(**FCH\_1 **==** LOW **&&** FCH\_2 **==** LOW**)** **{**  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Press to set "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" HORIZONTAL "**);**  **if** **(**digitalRead**(**pin\_pulsador1**)** **==** 1**)** **{**  estado **=** 10**;**  delay**(**10**);**  primeraVez **=** 1**;**  **};**  **}**  **break;**  **case** 10**:** //inicio de calibración IMUS en horizontal  Serial1**.**println**(**"Estado 10: Iniciando Calib Horizontal"**);**  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Calibration "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" in progress... "**);**  estado **=** 1**;**  **break;**  **case** 1**:**  Serial1**.**println**(**"Estado 1"**);**  init\_imus**();**  //Preparamos para lecturas comprobación imu  timer **=** millis**();**  comprPitch\_izq **=** 0**;**  comprRoll\_izq **=** 0**;**  comprPitch\_der **=** 0**;**  comprRoll\_der **=** 0**;**  contadorEstabilizacion **=** 0**;**  estado **=** 11**;**  **break;**  **case** 11**:** //Comprobacion de offset  ////Serial1.println("Estado 11");  FCH\_1 **=** digitalRead**(**FCH1**);**  FCH\_2 **=** digitalRead**(**FCH2**);**  //se realiza la calibración con lecturas a 50Hz ya que es  //la frecuencia a la que trabajan las IMUS  **if** **((**millis**()** **-** timer**)** **>=** 20**)** // Main loop runs at 50Hz  **{**  **if** **(**FCH\_1 **==** LOW **&&** FCH\_2 **==** LOW**)** **{**  //aseguramos que los zapatos están en zapatero horizontal  calculosIMU**();**  contadorEstabilizacion**++;**  **if** **(**contadorEstabilizacion **>** 100**)** **{**  **if** **(**caraZowi**)** **{**  comprPitch\_der **+=** pitch**;**  comprPitch\_izq **+=** pitchd**;**  comprRoll\_der **+=** roll**;**  comprRoll\_izq **+=** rolld**;**  **}**  **else** **{**  comprPitch\_der **+=** pitchd**;**  comprPitch\_izq **+=** pitch**;**  comprRoll\_der **+=** rolld**;**  comprRoll\_izq **+=** roll**;**  **}**  **}**  **if** **(**contadorEstabilizacion **>** 150**)** **{**  comprPitch\_der **=** abs**(**ToDeg**(**comprPitch\_der**)** **/** 50**);**  comprPitch\_izq **=** abs**(**ToDeg**(**comprPitch\_izq**)** **/** 50**);**  comprRoll\_izq **=** abs**(**ToDeg**(**comprRoll\_izq**)** **/** 50**);**  comprRoll\_der **=** abs**(**ToDeg**(**comprRoll\_der**)** **/** 50**);**  **if** **((**comprPitch\_der **>** umbralError**)** **||** **(**comprPitch\_izq **>** umbralError**)** **||** **(**comprRoll\_der **>** umbralError**)** **||** **(**comprRoll\_izq **>** umbralError**))** //Media  **{** //Error en offset imus  estado **=** 12**;**  Serial1**.**println**(**"ERROR. > a ESTADO 12 "**);**  //delay(1000);  **}**  **else** **{** //Offset en 0º IMUs OK  estado **=** 20**;**  nuevo0\_der **=** comprPitch\_der**;**  nuevo0\_izq **=** comprPitch\_izq**;**  Serial1**.**println**(**"Nuevo 0:"**);**  Serial1**.**println**(**nuevo0\_der**);**  Serial1**.**println**(**nuevo0\_izq**);**  **}**  **}**  **}**  **else** **{** //Zapato no está en zapatero horizontal  Serial1**.**println**(**"Zapatos mal puestos en posicion Horizontal"**);**  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Place shoes in "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" Horizontal box "**);**  delay**(**1000**);**  estado **=** 14**;**  **}**  **}**  **break;**  **case** 12**:** //Estado error en offset 0º de imus No se han calibrado correctamente  sonidoNOK**();**  Serial1**.**println**(**"Repite Posicion Horizontal"**);**  lcd**.**clear**();**  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Process failed "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" Retry "**);**  delay**(**1000**);**  estado **=** 14**;**  **break;**  **case** 20**:** //Estado calibración 0ºok, pasamos a calibración vertical  sonidoOK**();**  Serial1**.**println**(**"Calibracion IMU Horizontal ok"**);**  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Place shoes in "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" Vertical box "**);**  contadorEstabilizacion **=** 0**;**  estado **=** 21**;** //zapatos a 90 "calibracion"  timer **=** millis**();**  timerLed **=** millis**();**  delay**(**20**);**  counter **=** 0**;**  flagBoton **=** 0**;**  **break;**  **case** 21**:** //Estado para offset de 90º(calibración zapatos en vertical)  FCV\_1 **=** digitalRead**(**FCV1**);**  FCV\_2 **=** digitalRead**(**FCV2**);**  **if** **(**FCV\_1 **==** LOW **&&** FCV\_2 **==** LOW **&&** flagBoton **==** 0**)** **{**  //los zapatos deben estar insertados  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Press to set "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" VERTICAL "**);**  **};**  **if** **((**FCV\_1 **==** HIGH **||** FCV\_2 **==** HIGH**)** **&&** flagBoton **==** 0**)** **{**  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Place shoes in "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" Vertical box "**);**  **};**  **if** **((**millis**()** **-** timer**)** **>=** 20**)** // Main loop runs at 50Hz  //calibración zapatos en vertical  **{**  calculosIMU**();**  **if** **(**digitalRead**(**pin\_pulsador1**)** **==** 1**)** //esperamos pulsación para calibrar  **{**  flagBoton **=** 1**;**  **}**  **if** **(**flagBoton **==** 1**)** **{**  **if** **(**FCV\_1 **==** LOW **&&** FCV\_2 **==** LOW**)** **{** //Zapatos en zapatero vertical  contadorEstabilizacion**++;**  // lcd.clear();  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Calibration "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" in progress... "**);**  **if** **(**contadorEstabilizacion **>** 50**)** //esperamos a que se estabilice  **{**  **if** **(**caraZowi**)** **{**  comprRoll\_izq **+=** rolld**;**  comprRoll\_der **+=** roll**;**  **}**  **else** **{**  comprRoll\_izq **+=** roll**;**  comprRoll\_der **+=** rolld**;**  **}**  **}**  **if** **(**contadorEstabilizacion **>** 100**)**  //se han obtenido 50 medidas tras estabilizar  **{**  comprRoll\_izq **=** ToDeg**(**comprRoll\_izq **/** 50**);**  comprRoll\_der **=** ToDeg**(**comprRoll\_der **/** 50**);**  nuevo90\_izq **=** comprRoll\_izq**;**  nuevo90\_der **=** comprRoll\_der**;**  Serial1**.**println**(**"Calibracion IMU Vertical Ok"**);**  //calibración vertical hecha  //pasamos a colocar zapatos  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**"Connect+ON ZOWI "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**"Put shoes+press "**);**  Serial1**.**println**(**"Nuevo 90:"**);**  Serial1**.**println**(**nuevo90\_der**);**  Serial1**.**println**(**nuevo90\_izq**);**  flagBoton **=** 0**;**  estado **=** 2**;**  Serial1**.**println**(**"Coloca zap, enchufa y pulsa  boton para ajustar "**);**  sonidoOK**();**  timer **=** millis**();**  timerLed **=** millis**();**  delay**(**20**);**  counter **=** 0**;**  //Serial1.println("Aqui1");  **}**  **}**  **else** **{** //Zapatos no están en zapatero vertical  Serial1**.**println**(**"Zapatos mal colocados en posicion vertical"**);**  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Place shoes in "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" Vertical box "**);**  delay**(**1000**);**  lcd**.**clear**();**  estado **=** 21**;**  **}**  **}**  **}**  **break;**  **case** 2**:**  Serial1**.**println**(**"case2"**);**  // el led parpadea, se espera que se coloquen los zapatos  // en los pies de ZOWI.  //Salimos del estado actuando sobre el pulsador.  **if** **((**millis**()** **-** timer**)** **>=** 20**)** // Main loop runs at 50Hz  **{**  calculosIMU**();**  **}**  parpadeoLed**(**1**);** //sucede cada 1\*500ms  **if** **(**digitalRead**(**pin\_pulsador1**)** **==** 1**)** //AQUIIIIIII!!!!!  **{** //Arrancamos calibracion con el boton  parpadeoLed**(**0**);**  estado **=** 22**;**  cadera\_izq\_OK **=** 0**;**  cadera\_der\_OK **=** 0**;**  pie\_izq\_OK **=** 0**;**  pie\_der\_OK **=** 0**;**  num\_vueltas **=** 0**;**  digitalWrite**(**Led\_rojo**,** LOW**);**  digitalWrite**(**Led\_verde**,** LOW**);**  Serial1**.**println**(**"Iniciando comunicacion con Zum Zowi..."**);**  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Checking "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" connection... "**);**  **};**  **break;**  **case** 22**:**  Serial1**.**println**(**"case22"**);**  **if** **((**millis**()** **-** timer**)** **>=** 20**)** // Main loop runs at 50Hz  **{**  calculosIMU**();**  **}**  **if** **(**flag\_espera\_zum **==** 0**)** **{**  timer\_Zum **=** millis**();**  flag\_espera\_zum **=** 1**;**  **while** **(**Serial**.**available**()** **>=** 1**)** **{**  Serial1**.**println**(**Serial**.**read**());**  **}**  WriteComm**(**0**);** //Inicio comunicación Raspberry - ZUM mandamos IZUM  **}**  **if** **(**Serial**.**available**()** **>=** 1**)** **{**  lectura\_RB **=** Serial**.**read**();**  //Serial1.println(lectura\_RB);  **if** **(**lectura\_RB **==** 'B'**)** **{**  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**"Communication OK"**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**"Calibrating... "**);**  Serial1**.**println**(**"Comunicacion establecida."**);**  Serial1**.**println**(**"Calibrando..."**);**  estado **=** 3**;**  flag\_espera\_zum **=** 0**;**  **while** **(**Serial**.**available**()** **>=** 1**)** **{**  Serial1**.**println**(**Serial**.**read**());**  **}** //vaciamos serial  **break;**  **}**  **else** **if** **(**lectura\_RB **==** 'M'**)** **{**  //si la RPI envía una M indica que no se  //ha cargado el programa de calibración en Zowi  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**"Connection fault"**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" Check ZOWI "**);**  estado **=** 2**;**  flag\_espera\_zum **=** 0**;**  Serial1**.**println**(**"MAL. No cargada ZUM"**);**  Serial1**.**println**(**"Leyendo serial (3 veces):"**);**  //para vaciar últimos datos del serial  Serial1**.**println**(**Serial**.**read**());**  Serial1**.**println**(**Serial**.**read**());**  Serial1**.**println**(**Serial**.**read**());**  **}**  **}**  **else** **if** **(((**millis**()** **-** timer\_Zum**)** **>=** 40000**))** **{**  //si no responde en x secs: volver a intentar  Serial1**.**println**(**"TIMEOUT"**);**  Serial1**.**println**(**"Pulsa para volver a intentar..."**);**  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Timeout. Check "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" Zowi and push "**);**  estado **=** 2**;**  flag\_espera\_zum **=** 0**;**  **}**  **break;**  **case** 3**:** //Estado calibración articulaciones:  // Los zapatos ya están colocados en los pies y se comienza  // a ajustar offset de los servos.  parpadeoLed**(**4**);**  **if** **((**millis**()** **-** timer**)** **>=** 20**)** // Main loop runs at 50Hz  **{**  calculosIMU**();**  //Calculo errores de posicion respecto a los 0º y los 90º  //ajustados para el banco  **if** **(**caraZowi**)** **{** //dependiendo si está puesto el zowi de cara o de culo  err\_pie\_izq **=** nuevo90\_izq **-** ToDeg**(**rolld**);**  err\_cadera\_izq **=** nuevo0\_izq **-** ToDeg**(**pitchd**);**  err\_pie\_der **=** nuevo90\_der **-** ToDeg**(**roll**);**  err\_cadera\_der **=** nuevo0\_der **-** ToDeg**(**pitch**);**  **}**  **else** **{**  err\_pie\_izq **=** nuevo90\_izq **-** ToDeg**(**roll**);**  err\_cadera\_izq **=** nuevo0\_izq **-** ToDeg**(**pitch**);**  err\_pie\_der **=** nuevo90\_der **-** ToDeg**(**rolld**);**  err\_cadera\_der **=** nuevo0\_der **-** ToDeg**(**pitchd**);**  **}**  //Evaluamos error  **if** **(**abs**(**err\_cadera\_izq**)** **>** ErrorArtMAX**)** **{**  cadera\_izq\_OK **=** 0**;**  **}**  **else** **{**  cadera\_izq\_OK **=** 1**;**  **}**  **if** **(**abs**(**err\_pie\_izq**)** **>** ErrorArtMAX**)** **{**  pie\_izq\_OK **=** 0**;**  **}**  **else** **{**  pie\_izq\_OK **=** 1**;**  **}**  **if** **(**abs**(**err\_cadera\_der**)** **>** ErrorArtMAX**)** **{**  cadera\_der\_OK **=** 0**;**  **}**  **else** **{**  cadera\_der\_OK **=** 1**;**  **}**  **if** **(**abs**(**err\_pie\_der**)** **>** ErrorArtMAX**)** **{**  pie\_der\_OK **=** 0**;**  **}**  **else** **{**  pie\_der\_OK **=** 1**;**  **}**  num\_vueltas**++;** //para case de calibra servos  **if** **(!(**cadera\_der\_OK **==** 1 **&&** cadera\_izq\_OK **==** 1 **&&**  pie\_der\_OK **==** 1 **&&** pie\_izq\_OK **==** 1**))** **{**  //Si articulaciones con error de posicion->calibramos  **if** **(**num\_vueltas **<=** 330**)**  //Para 5 iteraciones de calibración en servos se llega a 330  **{**  calibra\_servos**();**  **}**  **else** **{**  estado **=** 40**;**  **};**  **}**  **else** **{**  estado **=** 40**;**  **};**  **}**  **break;**  **case** 40**:** //Estado chequeo de la calibracion  num\_vueltas **=** 0**;**  Serial**.**read**();**  Serial**.**read**();**  Serial**.**read**();**  Serial**.**read**();**  **if** **(**cadera\_der\_OK**==**1 **&&** cadera\_izq\_OK**==**1 **&&** pie\_der\_OK**==**1 **&&** pie\_izq\_OK**==**1**)** **{**  //Calibracion articulaciones OK  resultado **=** '1'**;**  result **=** 1**;**  // parpadeoLed(0);  // digitalWrite(Led\_rojo,LOW);  // digitalWrite(Led\_verde, HIGH);  // sonidoOK();  Serial1**.**println**(**"Calibración OK"**);**  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" Calibration OK "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" Writing EEPROM "**);**  cadera\_der\_OK **=** 0**;**  cadera\_izq\_OK **=** 0**;**  pie\_der\_OK **=** 0**;**  pie\_izq\_OK **=** 0**;**  dataError**[**0**]** **=** err\_cadera\_izq**;**  dataError**[**1**]** **=** err\_cadera\_der**;**  dataError**[**2**]** **=** err\_pie\_izq**;**  dataError**[**3**]** **=** err\_pie\_der**;**  WriteComm**(**7**);** //Comando WERC: errores de calibracion para RB)  data**[**0**]** **=** pos\_old\_cadera\_der**;**  data**[**1**]** **=** pos\_old\_pie\_der**;**  data**[**2**]** **=** pos\_old\_cadera\_izq**;**  data**[**3**]** **=** pos\_old\_pie\_izq**;**  WriteComm**(**22**);**  //Comando WOFF Pasamos a Zowi las posiciones HOME  delay**(**300**);**  data**[**0**]** **=** pos\_old\_cadera\_der**;**  data**[**1**]** **=** pos\_old\_pie\_der**;**  data**[**2**]** **=** pos\_old\_cadera\_izq**;**  data**[**3**]** **=** pos\_old\_pie\_izq**;**  WriteComm**(**2**);**  delay**(**300**);**  WriteComm**(**10**);** //Comando WSQL Salva base datos  Serial1**.**print**(**"Cadera Izq :"**);**  Serial1**.**println**(**pos\_old\_cadera\_izq**);**  Serial1**.**print**(**"Pie Izq :"**);**  Serial1**.**println**(**pos\_old\_pie\_izq**);**  Serial1**.**print**(**"Cadera Der :"**);**  Serial1**.**println**(**pos\_old\_cadera\_der**);**  Serial1**.**print**(**"Pie Der :"**);**  Serial1**.**println**(**pos\_old\_pie\_der**);**  Serial1**.**print**(**"Error cadera izq :"**);**  Serial1**.**println**(**err\_cadera\_izq**);**  Serial1**.**print**(**"Error pie izq :"**);**  Serial1**.**println**(**err\_pie\_izq**);**  Serial1**.**print**(**"Error cadera der :"**);**  Serial1**.**println**(**err\_cadera\_der**);**  Serial1**.**print**(**"Error pie der :"**);**  Serial1**.**println**(**err\_pie\_der**);**  delay**(**500**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**"Loading Test Prg"**);**  WriteComm**(**8**);** //Comando FZUM Cargar programa Test  lectura\_RB **=** 0**;**  timer\_WD **=** millis**();**  **do** **{**  **if** **(**Serial**.**available**()** **>=** 1**)** **{**  lectura\_RB **=** Serial**.**read**();**  Serial1**.**println**(**lectura\_RB**);**  **}**  **if** **((**millis**()** **-** timer\_WD**)** **>=** 20000**)** **{**  sonidoNOK**();**  parpadeoLed**(**0**);**  digitalWrite**(**Led\_rojo**,** HIGH**);**  digitalWrite**(**Led\_verde**,** LOW**);**  **break;**  **}**  **}** **while** **((**lectura\_RB **!=** 'M'**)** **&&** **(**lectura\_RB **!=** 'B'**));**  //esperamos a que la RPI envíe B o M  //dependiendo de carga programa Test  **if** **(**lectura\_RB **==** 'B'**)** **{**  sonidoOK**();**  parpadeoLed**(**0**);**  digitalWrite**(**Led\_rojo**,** LOW**);**  digitalWrite**(**Led\_verde**,** HIGH**);**  **}**  **else** **if** **(**lectura\_RB **==** 'M'**)** **{**  sonidoNOK**();**  parpadeoLed**(**0**);**  digitalWrite**(**Led\_rojo**,** HIGH**);**  digitalWrite**(**Led\_verde**,** LOW**);**  **}**  lectura\_RB **=** 0**;**  Serial**.**read**();**  Serial**.**read**();**  Serial**.**read**();**  Serial**.**read**();**  delay**(**1000**);**  estado **=** 4**;** //Realizar otra calibración si pulsador  **}**  **else** **{** //La calibracion de alguna articulacion ha fallado  sonidoNOK**();**  resultado **=** '0'**;**  result **=** 0**;**  parpadeoLed**(**0**);**  digitalWrite**(**Led\_rojo**,** HIGH**);**  digitalWrite**(**Led\_verde**,** LOW**);**  dataError**[**0**]** **=** err\_cadera\_izq**;**  dataError**[**1**]** **=** err\_cadera\_der**;**  dataError**[**2**]** **=** err\_pie\_izq**;**  dataError**[**3**]** **=** err\_pie\_der**;**  WriteComm**(**7**);** //Comando WERC: errores de calibracion para RB)  data**[**0**]** **=** 90**;**  data**[**1**]** **=** 90**;**  data**[**2**]** **=** 90**;**  data**[**3**]** **=** 90**;** //si sale mal, mandamos 90 En BBDD = 0 (90-90)  WriteComm**(**2**);** //Comando WOFF Pasamos a Zowi las posiciones HOME  WriteComm**(**10**);** //Salva base datos  //lcd.clear();  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**" CALIBRATION "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" FAILED "**);**  Serial1**.**println**(**"Calibracion MAL"**);**  Serial1**.**println**(**"Error calibracion articulaciones"**);**  Serial1**.**print**(**"Error cadera izq :"**);**  Serial1**.**println**(**err\_cadera\_izq**);**  Serial1**.**print**(**"Error pie izq :"**);**  Serial1**.**println**(**err\_pie\_izq**);**  Serial1**.**print**(**"Error cadera der :"**);**  Serial1**.**println**(**err\_cadera\_der**);**  Serial1**.**print**(**"Error pie der :"**);**  Serial1**.**println**(**err\_pie\_der**);**  delay**(**1000**);**  estado **=** 4**;** //Realizar otra calibración si pulsador  **}**  **break;**  **case** 4**:** //Estado otra calibracion  num\_vueltas **=** 0**;**  lcd**.**setCursor**(**0**,** 0**);**  lcd**.**print**(**"Press for a new "**);**  lcd**.**setCursor**(**0**,** 1**);**  lcd**.**print**(**" calibration "**);**  **if** **(**digitalRead**(**pin\_pulsador1**)** **==** 1**)**  //pasamos a realizar otra calibración al pulsar  **{**  Serial1**.**println**(**"Iniciamos otra calibracion 4 a 2"**);**  estado **=** 2**;**  timer **=** millis**();**  timerLed **=** millis**();**  delay**(**20**);**  counter **=** 0**;**  **}**  **break;**  **}**  **}**  void calibra\_servos**()**  **{**  **switch** **(**num\_vueltas**)** **{**  **case** 50**:**  data**[**0**]** **=** 3**;** //Indicador cadera izquierda  data**[**1**]** **=** 90**;**  data**[**2**]** **=** 0**;**  WriteComm**(**6**);** //MsXC Comando mover servo especifico a posicion especifica  pos\_old\_cadera\_izq **=** 90**;**  **break;**  **case** 60**:**  data**[**0**]** **=** 4**;** //Indicador pie izquierdo  data**[**1**]** **=** 90**;**  data**[**2**]** **=** 0**;**  WriteComm**(**6**);** //MsXC Comando mover servo especifico a posicion especifica  pos\_old\_pie\_izq **=** 90**;**  **break;**  **case** 70**:**  data**[**0**]** **=** 1**;** //Indicador cadera derecha  data**[**1**]** **=** 90**;**  data**[**2**]** **=** 0**;**  WriteComm**(**6**);** //MsXC Comando mover servo especifico a posicion especifica  pos\_old\_cadera\_der **=** 90**;**  **break;**  **case** 80**:**  data**[**0**]** **=** 2**;** //Indicador pie derecho  data**[**1**]** **=** 90**;**  data**[**2**]** **=** 0**;**  WriteComm**(**6**);** //MsXC Comando mover servo especifico a posicion especifica  pos\_old\_pie\_der **=** 90**;**  **break;**  **case** 100**:**  **case** 150**:**  **case** 200**:**  **case** 250**:**  **case** 300**:**  data**[**0**]** **=** 3**;** //Indicador cadera izquierda  data**[**1**]** **=** round**(**pos\_old\_cadera\_izq **+** err\_cadera\_izq**);**  data**[**2**]** **=** 0**;**  WriteComm**(**6**);** //MsXC Comando mover servo especifico a posicion especifica  pos\_old\_cadera\_izq **=** round**(**pos\_old\_cadera\_izq **+** err\_cadera\_izq**);**  //Redondeamos en lugar de cast para mejorar el error [servo.write(int)]  **break;**  **case** 110**:**  **case** 160**:**  **case** 210**:**  **case** 260**:**  **case** 310**:**  data**[**0**]** **=** 4**;** //Indicador pie izquierdo  **if** **(**caraZowi**)** **{**  data**[**1**]** **=** round**(**pos\_old\_pie\_izq **+** err\_pie\_izq**);**  **}**  **else** **{**  data**[**1**]** **=** round**(**pos\_old\_pie\_izq **-** err\_pie\_izq**);**  **}**  data**[**2**]** **=** 0**;**  WriteComm**(**6**);** //Comando mover servo especifico a posicion especifica  **if** **(**caraZowi**)** **{**  pos\_old\_pie\_izq **=** round**(**pos\_old\_pie\_izq **+** err\_pie\_izq**);**  **}**  **else** **{**  pos\_old\_pie\_izq **=** round**(**pos\_old\_pie\_izq **-** err\_pie\_izq**);**  **}**  **break;**  **case** 120**:**  **case** 170**:**  **case** 220**:**  **case** 270**:**  **case** 320**:**  data**[**0**]** **=** 1**;** //Indicador pie izquierdo  data**[**1**]** **=** round**(**pos\_old\_cadera\_der **+** err\_cadera\_der**);**  data**[**2**]** **=** 0**;**  WriteComm**(**6**);** //MsXCComando mover servo especifico a posicion especifica  pos\_old\_cadera\_der **=** round**(**pos\_old\_cadera\_der **+** err\_cadera\_der**);**  **break;**  **case** 130**:**  **case** 180**:**  **case** 230**:**  **case** 280**:**  **case** 330**:**  data**[**0**]** **=** 2**;** //Indicador pie derecho  **if** **(**caraZowi**)** **{**  data**[**1**]** **=** round**(**pos\_old\_pie\_der **+** err\_pie\_der**);**  **}**  **else** **{**  data**[**1**]** **=** round**(**pos\_old\_pie\_der **-** err\_pie\_der**);**  **}**  data**[**2**]** **=** 0**;**  WriteComm**(**6**);** //MsXC Comando mover servo especifico a posicion especifica  **if** **(**caraZowi**)** **{**  pos\_old\_pie\_der **=** round**(**pos\_old\_pie\_der **+** err\_pie\_der**);**  **}**  **else** **{**  pos\_old\_pie\_der **=** round**(**pos\_old\_pie\_der **-** err\_pie\_der**);**  **}**  **break;**  **default:**  **break;**  **}**  **}**  void parpadeoLed**(**int freqLed**)**  **{**  **if** **(**freqLed **==** 0**)** **{**  digitalWrite**(**Led\_azul**,** LOW**);**  **}**  **else** **if** **((**millis**()** **-** timerLed**)** **>=** **(**500 **/** freqLed**))**  // Led parpadea N vez por segundo, N = freqLed  **{**  timerLed **=** millis**();**  digitalWrite**(**Led\_azul**,** **!**digitalRead**(**Led\_azul**));**  **}**  **}**  void calculosIMU**()**  **{**  counter**++;**  timer\_old **=** timer**;**  timer **=** millis**();**  **if** **(**timer **>** timer\_old**)**  G\_Dt **=** **(**timer **-** timer\_old**)** **/** 1000.0**;**  // Real time of loop run. We use this on the DCM algorithm  // (gyro integration time)  **else**  G\_Dt **=** 0**;**  // \*\*\* DCM algorithm  // Data adquisition  Read\_Gyro**();** // This read gyro data  Read\_Accel**();** // Read I2C accelerometer  **if** **(**counter **>** 5**)** // Read compass data at 10Hz... (5 loop runs)  **{**  counter **=** 0**;**  Read\_Compass**();** // Read I2C magnetometer  Compass\_Heading**();** // Calculate magnetic heading  **}**  // Calculations...  Matrix\_update**();**  Normalize**();**  Drift\_correction**();**  Euler\_angles**();**  // \*\*\*  //printdata(); //Muestra datos leidos IMU  **}**  int init\_imus**()**  **{**  Serial1**.**println**(**"Calibrando imus..."**);**  digitalWrite**(**STATUS\_LED**,** LOW**);**  delay**(**1500**);**  Accel\_Init**();**  Compass\_Init**();**  Gyro\_Init**();**  delay**(**20**);**  **for** **(**int i **=** 0**;** i **<** 32**;** i**++)** // We take some readings...  **{**  Read\_Gyro**();**  Read\_Accel**();**  **for** **(**int y **=** 0**;** y **<** 6**;** y**++)** // Cumulate values  **{**  AN\_OFFSET**[**y**]** **+=** AN**[**y**];**  AN\_OFFSETd**[**y**]** **+=** ANd**[**y**];**  **}**  delay**(**20**);**  **}**  **for** **(**int y **=** 0**;** y **<** 6**;** y**++)** **{**  AN\_OFFSET**[**y**]** **=** AN\_OFFSET**[**y**]** **/** 32**;**  AN\_OFFSETd**[**y**]** **=** AN\_OFFSETd**[**y**]** **/** 32**;**  **}**  AN\_OFFSET**[**5**]** **-=** GRAVITY **\*** SENSOR\_SIGN**[**5**];**  AN\_OFFSETd**[**5**]** **-=** GRAVITY **\*** SENSOR\_SIGNd**[**5**];**  delay**(**4000**);**  digitalWrite**(**STATUS\_LED**,** HIGH**);**  timer **=** millis**();**  delay**(**20**);**  counter **=** 0**;**  **}**  int WriteComm**(**int command**)**  **{**  int ind **=** 0**;**  **switch** **(**command**)** **{**  **case** 0**:** //Iniciar comunicación con ZUM.  **for** **(**int i **=** 0**;** i **<** 30**;** i**++)** **{**  trama**[**i**]** **=** 0**;**  **}**  **for** **(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  trama**[**i**]** **=** C0**[**i**];**  **}**  trama**[**6**]** **=** '#'**;**  Serial**.**write**(**trama**);**  Serial**.**flush**();**  Serial1**.**println**(**trama**);**  **break;**  **case** 8**:** //cargar programa test y guardar en base de datos.  **for** **(**int i **=** 0**;** i **<** 30**;** i**++)** **{**  trama**[**i**]** **=** 0**;**  **}**  **for** **(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  trama**[**i**]** **=** C8**[**i**];**  **}**  trama**[**6**]** **=** '#'**;**  Serial**.**write**(**trama**);**  **;**  Serial1**.**println**(**trama**);**  **break;**  **case** 1**:** //Leer offset  delay**(**100**);**  ind **=** 0**;**  //Quedamos a espera de la lectura  **while** **(**Wire**.**available**())** **{**  data**[**ind**]** **=** Wire**.**read**();**  ind**++;**  **}**  dataINT**[**0**]** **=** data**[**0**];**  dataINT**[**1**]** **=** data**[**1**];**  dataINT**[**2**]** **=** data**[**2**];**  dataINT**[**3**]** **=** data**[**3**];**  Serial1**.**println**(**"Posicion HOME leida de ZOWI:"**);**  Serial1**.**print**(**"Home cadera izq: "**);**  Serial1**.**println**(**dataINT**[**0**]);**  Serial1**.**print**(**"Home pie izq: "**);**  Serial1**.**println**(**dataINT**[**1**]);**  Serial1**.**print**(**"Home cadera der: "**);**  Serial1**.**println**(**dataINT**[**2**]);**  Serial1**.**print**(**"Home pie der: "**);**  Serial1**.**println**(**dataINT**[**3**]);**  **break;**  **case** 22**:** //Escribir offset WOFC Izquierda  **for** **(**int i **=** 0**;** i **<** 29**;** i**++)** **{**  trama**[**i**]** **=** 0**;**  **}**  ///////////////////////////////////  /\* INFORMACION  data[0]=pos\_old\_cadera\_der; //COLUMNA 1 para BASE DATOS  data[1]=pos\_old\_pie\_der; //COLUMNA 3 oara BASE DATOS  data[2]=pos\_old\_cadera\_izq; //COLUMNA 0 oara BASE DATOS  data[3]=pos\_old\_pie\_izq; //COLUMNA 2 para base DATOS  \*/ //////////////////////////////////////////////////  **for** **(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  trama**[**i**]** **=** C2**[**i**];**  **}**  trama**[**6**]** **=** '2'**;**  numeroData **=** 2**;**  **for** **(**int j **=** 7**;** j **<** 19**;)** **{**  str **=** String**(**int**(**data**[**numeroData**]** **/** 100**));**  str**.**toCharArray**(**temp**,** 2**);**  trama**[**j**]** **=** temp**[**0**];**  j**++;**  str **=** String**(**int**((**data**[**numeroData**]** **%** 100**)** **/** 10**));**  str**.**toCharArray**(**temp**,** 2**);**  trama**[**j**]** **=** temp**[**0**];**  j**++;**  str **=** String**(**int**(**data**[**numeroData**]** **%** 10**));**  str**.**toCharArray**(**temp**,** 2**);**  trama**[**j**]** **=** temp**[**0**];**  numeroData**++;**  **if** **(**numeroData **==** 4**)** **{**  numeroData **=** 0**;**  **};**  j**++;**  //Serial1.println("aqui");  **}**  trama**[**19**]** **=** '#'**;**  Serial1**.**println**(**trama**);**  //Mandamos los valores  Serial**.**write**(**trama**);**  **break;**  **case** 2**:** //Escribir offset WOFC  **for** **(**int i **=** 0**;** i **<** 29**;** i**++)** **{**  trama**[**i**]** **=** 0**;**  **}**  ///////////////////////////////////  /\* INFORMACION  data[0]=pos\_old\_cadera\_der; //COLUMNA 1 para BASE DATOS  data[1]=pos\_old\_pie\_der; //COLUMNA 3 oara BASE DATOS  data[2]=pos\_old\_cadera\_izq; //COLUMNA 0 oara BASE DATOS  data[3]=pos\_old\_pie\_izq; //COLUMNA 2 para base DATOS  \*/ //////////////////////////////////////////////////  **for** **(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  trama**[**i**]** **=** C2**[**i**];**  **}**  trama**[**6**]** **=** '1'**;**  numeroData **=** 0**;**  **for** **(**int z **=** 7**;** z **<** 19**;)** **{**  str **=** String**(**int**(**data**[**numeroData**]** **/** 100**));**  str**.**toCharArray**(**temp**,** 2**);**  trama**[**z**]** **=** temp**[**0**];**  z**++;**  str **=** String**(**int**((**data**[**numeroData**]** **%** 100**)** **/** 10**));**  str**.**toCharArray**(**temp**,** 2**);**  trama**[**z**]** **=** temp**[**0**];**  z**++;**  str **=** String**(**int**(**data**[**numeroData**]** **%** 10**));**  str**.**toCharArray**(**temp**,** 2**);**  trama**[**z**]** **=** temp**[**0**];**  numeroData**++;**  z**++;**  //Serial1.println("aqui");  **}**  trama**[**19**]** **=** '#'**;**  Serial1**.**println**(**trama**);**  //Mandamos los valores  Serial**.**write**(**trama**);**  **break;**  **case** 3**:** //Mover servos a 90º  **for** **(**int i **=** 0**;** i **<** 30**;** i**++)** **{**  trama**[**i**]** **=** 0**;**  **}**  **for** **(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  trama**[**i**]** **=** C3**[**i**];**  **}**  trama**[**6**]** **=** '#'**;**  Serial**.**write**(**trama**);**  Serial1**.**println**(**"Enviado M90C:"**);**  **break;**  **case** 4**:** //Mover servos a HOME (90 calibrado)  **for** **(**int i **=** 0**;** i **<** 30**;** i**++)** **{**  trama**[**i**]** **=** 0**;**  **}**  **for** **(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  trama**[**i**]** **=** C4**[**i**];**  **}**  trama**[**6**]** **=** '#'**;**  Serial**.**write**(**trama**);**  **break;**  **case** 5**:** //Mover 4 servos a posicion especifica (para calibracion con IMU)  Serial1**.**println**(**"Enviado MSSC:"**);**  //Mandamos los valores  //Wire.write(data);  **break;**  **case** 6**:** //Mover un servo a posicion especifica (para calibracion con IMU)MSxC  **for** **(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  trama**[**i**]** **=** C6**[**i**];**  **}**  //Serial1.println(int(data[1]));  str **=** String**(**int**(**data**[**0**]));**  str**.**toCharArray**(**temp**,** 2**);**  trama**[**6**]** **=** temp**[**0**];**  str **=** String**(**int**(**data**[**1**]** **/** 100**));**  str**.**toCharArray**(**temp**,** 2**);**  trama**[**7**]** **=** temp**[**0**];**  str **=** String**(**int**((**data**[**1**]** **%** 100**)** **/** 10**));**  str**.**toCharArray**(**temp**,** 2**);**  trama**[**8**]** **=** temp**[**0**];**  str **=** String**(**int**(**data**[**1**]** **%** 10**));**  str**.**toCharArray**(**temp**,** 2**);**  trama**[**9**]** **=** temp**[**0**];**  trama**[**10**]** **=** '#'**;**  Serial**.**write**(**trama**);**  ////Serial1.write(trama);  Serial1**.**println**(**trama**);**  **break;**  **case** 7**:** //WERC  // for (int i =0; i<=40;i++){  // tramaError[i]=0;  // }  **for** **(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  tramaError**[**i**]** **=** C7**[**i**];**  **}**  **for** **(**int i **=** 0**;** i **<** 4**;** i**++)** **{**  **if** **(**dataError**[**i**]** **<** 0**)** **{**  tramaError**[(**6 **+** i **\*** 8**)]** **=** '-'**;**  **}**  **else** **{**  tramaError**[(**6 **+** i **\*** 8**)]** **=** '+'**;**  **}**  dataError**[**i**]** **=** abs**(**int**(**dataError**[**i**]** **\*** 100**));**  str **=** String**(**int**(**dataError**[**i**]** **/** 10000**));**  str**.**toCharArray**(**temp**,** 2**);**  tramaError**[(**7 **+** i **\*** 8**)]** **=** temp**[**0**];**  str **=** String**((**int**(**dataError**[**i**]** **/** 1000**))** **%** 10**);**  str**.**toCharArray**(**temp**,** 2**);**  tramaError**[(**8 **+** i **\*** 8**)]** **=** temp**[**0**];**  str **=** String**((**int**(**dataError**[**i**]** **/** 100**)** **%** 10**));**  str**.**toCharArray**(**temp**,** 2**);**  tramaError**[(**9 **+** i **\*** 8**)]** **=** temp**[**0**];**  tramaError**[(**10 **+** i **\*** 8**)]** **=** '.'**;**  str **=** String**((**int**(**dataError**[**i**])** **/** 10**)** **%** 10**);**  str**.**toCharArray**(**temp**,** 2**);**  tramaError**[(**11 **+** i **\*** 8**)]** **=** temp**[**0**];**  str **=** String**((**int**(**dataError**[**i**])** **%** 10**));**  str**.**toCharArray**(**temp**,** 2**);**  tramaError**[(**12 **+** i **\*** 8**)]** **=** temp**[**0**];**  tramaError**[(**13 **+** i **\*** 8**)]** **=** '\*'**;**  **}**  tramaError**[**38**]** **=** resultado**;**  tramaError**[**39**]** **=** '#'**;**  Serial1**.**println**(**tramaError**);**  Serial**.**write**(**tramaError**);**  // for (int i =0; i<=40;i++){  // tramaError[i]=0;  // }  **break;**  **case** 9**:** //Apagar RB  **for** **(**int i **=** 0**;** i **<** 30**;** i**++)** **{**  trama**[**i**]** **=** 0**;**  **}**  **for** **(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  trama**[**i**]** **=** C9**[**i**];**  **}**  trama**[**6**]** **=** '#'**;**  Serial**.**write**(**trama**);**  **break;**  **case** 10**:**  **for** **(**int i **=** 0**;** i **<** 30**;** i**++)** **{**  trama**[**i**]** **=** 0**;**  **}**  **for** **(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  trama**[**i**]** **=** C10**[**i**];**  **}**  trama**[**6**]** **=** '#'**;**  Serial**.**write**(**trama**);**  **default:**  **break;**  **}**  data**[**0**]** **=** 0**;**  data**[**1**]** **=** 0**;**  data**[**2**]** **=** 0**;**  data**[**3**]** **=** 0**;**  data**[**4**]** **=** 0**;**  **return** 1**;**  **}**  void sonidoNOK**()**  **{**  tone**(**Pin\_buzz**,** 100**);**  delay**(**800**);**  noTone**(**Pin\_buzz**);**  **}**  void sonidoOK**()**  **{**  tone**(**Pin\_buzz**,** 400**);**  delay**(**300**);**  noTone**(**Pin\_buzz**);**  delay**(**50**);**  **}**  void sonidostart**()**  **{**  tone**(**Pin\_buzz**,** 560**);**  delay**(**300**);**  noTone**(**Pin\_buzz**);**  delay**(**10**);**  tone**(**Pin\_buzz**,** 545**);**  delay**(**150**);**  tone**(**Pin\_buzz**,** 575**);**  delay**(**400**);**  noTone**(**Pin\_buzz**);**  delay**(**50**);**  **}** |