

APÉNDICE D: Código en Arduino

Archivo: CONTROLADOR_PID.ino

```

1  #include <Wire.h> // Esta librería permite comunicar con dispositivos I2C TWI.
2  #include "Kalman.h" // Carga de la librería Filtro Kalman
3
4  Kalman kalmanX; //
5  Kalman kalmanY; //Aquí está creando los objetos kalmanX y KalmanY de tipo Kalman(librería). Más
6  adelante si quiero usar una función dentro de Kalman.h la usaré particularizada al objeto haciendo:
7  kalmanX.funcion()o kalmanY.funcion()
8
9  /* Datos de IMU */
10 double accX, accY, accZ;
11 double gyroX, gyroY, gyroZ;
12 int16_t tempRaw; //Aquí se almacenará la temperatura en crudo (no en °C)
13
14 double gyroXangle, gyroYangle; // Ángulo calculado utilizando sólo los giróscopos
15 double compAngleX, compAngleY; // Ángulo calculado utilizando un filtro complementario
16 double kalAngleX, kalAngleY; // Ángulo calculado utilizando un filtro Kalman
17
18 uint32_t timer;
19 uint8_t i2cData[14]; // Buffer para datos I2C. Creación del vector i2cData reservando 14 elementos
20
21
22 /* PARA LOS ENCODERS, VELOCIDAD Y ACELERACIÓN */
23 //Izquierdo
24 #define encoder0PinA 18
25 #define encoder0PinB 19
26 volatile float encoder0Pos = 0;
27 unsigned long time1;
28 float pos0;
29 float vel;
30 float vel0;
31 float acel;
32 float velf;
33
34 //Derecho
35 #define encoder1PinA 2
36 #define encoder1PinB 3
37 volatile float encoder1Pos = 0;
38 unsigned long time2;
39 float pos0d;
40 float veld;
41 float vel0d;
42 float aceld;
43 float velfd;
44
45 //////////////////////////////////////////////////
46
47 /* PARA EL MOTOR */
48
49 #define PinPWMMD 7
50 #define PinPWMI 9
51 #define PinIn1I 40
52 #define PinIn2I 41
53 #define PinIn1D 43
54 #define PinIn2D 42
55
56 float PWMMD;
57 float PWMI;
58
59 //////////////////////////////////////////////////
60
61 /* PARA EL CONTROL PID*/
62
63 #define Pinpot1 A2
64 #define Pinpot2 A1
65 #define Pinpot3 A0
66 int lectpotP;
67 int lectpotTi;
68 int lectpotTd;
69
70 float u;
71 float r;
72 float e;
73 float Kp;
74 float Ki;
75 float Kd;
76 float dInput, lastInput;
77 float ITerm;
78 float Ahora;
79 float Cambiot, Ultimot;
80
81 //////////////////////////////////////////////////
82
83 void setup() {
84     Serial.begin(115200);
85     Wire.begin();
86     TWBR = ((F_CPU / 400000L) - 16) / 2; // Configurar la frecuencia I2C a 400kHz TWBR - TWI Bit Rate
87     Register --> He econtrado esto: TWBR selects the division factor for the bit rate generator. The bit
88     rate generator is a frequency divider which generates the SCL clock frequency in the Master modes.
89     //En el Datasheet de la MPU6050 dicen que la máx. velocidad
90     del bus es 400kHz (por eso él lo pone a esta vel.)
91
92     i2cData[0] = 7; // Configurar la tasa de muestreo a 1000Hz -> 8kHz/(7+1) = 1000Hz

```

```

93 >>Config tasa muestreo. Registro 0x19hex=25dec (Nuestro giroscopo funciona a 8kHz, ver pag.12-
94 13 Mapa Registro pdf)
95 i2cData[1] = 0x00; // Deshabilitar FSYNC y configurar los filtros a 260 Hz Acel., 256 Hz Giros., y
96 muestreo 8 KHz >>Sincronización y Filtrado. Registro 0x1Ahex=26dec (ver pag.13 Mapa Registro pdf)
97 i2cData[2] = 0x00; // Configurar Rango Giroscopos a ±250deg/s
98 >>Configuración Giróscopos. Registro 0x1Bhex=27dec Lo pone a máxima sensibilidad/minimo rango y
99 deshabilita el autotest(ver pag.14 Mapa Registro pdf)
100 i2cData[3] = 0x00; // Configurar Rango Acelerómetros a ±2g
101 >>Configuración Acelerómetros. Registro 0x1Chex=28dec Lo pone a máxima sensibilidad/minimo rango y
102 deshabilita el autotest (ver pag.15-16 Mapa Registro pdf)
103 while (i2cWrite(0x19, i2cData, 4, false)); // >>Meter los anteriores 4 valores en cada posición del
104 registro y es un while porque devuelve un 0 cuando es con éxito y si no lo reintenta (ver archivo
105 i2C.ino)
106 while (i2cWrite(0x6B, 0x01, true)); // >>Deshabilitar modo standby. Usar como señal de reloj
107 el giroscopo del eje X. Esto se mete en el registro 0x6Bhex=107dec (ver pag.41 Mapa Registro pdf)
108
109 while (i2cRead(0x75, i2cData, 1)); //Leer los registros 0x75hex=117dec; y los mete en el vector
110 i2cData. Como el registro 0x75 es el último, el resto de valores del vector quedará con los valores
111 que tenia antes.
112 if (i2cData[0] != 0x68) { // Leer el registro "Quién Soy?" (ver pag. 46) devuelve el valor 0x68 para
113 decir cuál es la dirección del dispositivo (aunque realmente sabemos que la dirección será sólo 0x68
114 cuando el pin AD0 esté LOW y 0x69 cuando el pin AD0 esté HIGH. Esto aparece en el datasheet)
115 Serial.print(F("Error leyendo sensor"));
116 while (1);
117 }
118
119 delay(100); // Espera para que se establezca el sensor
120
121 /* Ángulos de inicio para el cálculo kalman y cálculo con giróscopos*/
122 while (i2cRead(0x3B, i2cData, 6)); //Lectura de los Registros de Acelerómetros:0x3Bhex=59dec
123 y 0x3Chex=60dec; 0x3Dhex=61dec y 0x3Ehex=62dec; 0x3Fhex=63dec y 0x40hex=64dec; y los mete en el
124 vector i2cData ver pag.30 Mapa Registro pdf)
125 accX = (i2cData[0] << 8) | i2cData[1]; // (i2cData[0] << 8)=esto es un desplazamiento de 8 bits a
126 la izquierda de forma que el i2cData[0] queda en las posiciones más significativas y a la derecha
127 quedan ceros.
128 accY = (i2cData[2] << 8) | i2cData[3]; // Con la operación | se suman bit a bit las posiciones
129 menos significativas con el valor i2cData[1]. Así se obtiene un número
130 accZ = (i2cData[4] << 8) | i2cData[5]; // de 16 bit a partir de dos números de 8 bit (porque los
131 registros sólo pueden guardar 8 bits)
132
133 // Source: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf eq. 25 and eq. 26
134 // atan2 devuelve valores de -π a π (radianes)
135 // Posteriormente se convierte de radianes a grados
136 double roll = atan2(accY, accZ) * RAD_TO_DEG;
137 double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
138
139 // Inicializar ángulos para Kalman, filtro complementario y giroscopos.
140 kalmanX.setAngle(roll);
141 kalmanY.setAngle(pitch);
142 gyroXangle = roll;
143 gyroYangle = pitch;
144 compAngleX = roll;
145 compAngleY = pitch;
146 timer = micros();
147
148 /////// CÓDIGO PARA LA POSICION, VELOCIDAD Y ACELERACIÓN /////
149 //izquierdo
150 pinMode(encoder0PinA, INPUT);
151 digitalWrite(encoder0PinA, HIGH); // Activar resistencias pullup
152 pinMode(encoder0PinB, INPUT);
153 digitalWrite(encoder0PinB, HIGH); // Activar resistencias pullup
154 //Recordar que son encoders de cuadratura, hay dos sensores por cada motor colocados a 90° uno del
155 otro
156 attachInterrupt(5, doEncoderA0, CHANGE); // Encoder A en la interrupción 5 (pin 18)
157 attachInterrupt(4, doEncoderB0, CHANGE); // Encoder B en la interrupción 4 (pin 19)
158
159 //Derecho
160 pinMode(encoder1PinA, INPUT);
161 digitalWrite(encoder1PinA, HIGH); // Activar resistencias pullup
162 pinMode(encoder1PinB, INPUT);
163 digitalWrite(encoder1PinB, HIGH); // Activar resistencias pullup
164 //Recordar que son encoders de cuadratura, hay dos sensores por cada motor colocados a 90° uno del
165 otro
166 attachInterrupt(0, doEncoderA1, CHANGE); // Encoder A en la interrupción 0 (pin 2)
167 attachInterrupt(1, doEncoderB1, CHANGE); // Encoder B en la interrupción 1 (pin 3)
168
169 time1=millis(); // Inicializamos tiempo y posicion para el calculo de velocidad
170 time2=millis(); // Inicializamos tiempo y posicion para el calculo de velocidad
171
172
173 ////////////// MOTOR ///////////////////
174 //Para subir la frecuencia del PWM pin 7 del Mega 2560 de 489Hz a 31333Hz
175 TCCR4B = TCCR4B & 0b00010000 | 0x01;
176 //Para subir la frecuencia del PWM pin 9 del Mega 2560 de 489Hz a 31333Hz
177 TCCR2B = TCCR2B & 0b00010000 | 0x01;
178
179 pinMode(PinPWMD, OUTPUT);
180 pinMode(PinPWMI, OUTPUT);
181 pinMode(PinIn1I, OUTPUT);
182 pinMode(PinIn2I, OUTPUT);
183 pinMode(PinIn1D, OUTPUT);
184 pinMode(PinIn2D, OUTPUT);
185
186 ///CONTROLADOR///
187 r=-1;
188 u=0;e=0;
189 ///////////////////////////////////////////////////
190
191 }
192

```

```

193 void loop() {
194
195 //CÓDIGO PARA LA IMU//
196
197 /* Actualización de todos los valores */
198 while (i2cRead(0x3B, i2cData, 14));
199 accX = ((i2cData[0] << 8) | i2cData[1]) - 432.0; //Se suma el offset para calibrado;
200 accY = ((i2cData[2] << 8) | i2cData[3]) - 291.16; //Se suma el offset para calibrado;
201 accZ = ((i2cData[4] << 8) | i2cData[5]) - 307.52; //Se suma el offset para calibrado;
202 tempRaw = (i2cData[6] << 8) | i2cData[7];
203 gyroX = (i2cData[8] << 8) | i2cData[9];
204 gyroY = (i2cData[10] << 8) | i2cData[11];
205 gyroZ = (i2cData[12] << 8) | i2cData[13];
206
207 double dt = (double)(micros() - timer) / 1000000; // Cálculo de dt
208 timer = micros();
209
210 // atan2 devuelve valores de -π a π (radianes)
211 // Posteriormente se convierte de radianes a grados
212
213 double roll = atan2(accY, accZ) * RAD_TO_DEG;
214 double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
215
216 double gyroXrate = gyroX / 131.0; // Conversión a °/s
217 double gyroYrate = gyroY / 131.0; // Conversión a °/s
218
219 // Esto soluciona el problema de transición cuando el ángulo de los acelerómetros salta de -180° a
220 180°
221 if ((roll < -90 && kalAngleX > 90) || (roll > 90 && kalAngleX < -90)) {
222     kalmanX.setAngle(roll);
223     compAngleX = roll;
224     kalAngleX = roll;
225     gyroXangle = roll;
226 } else
227     kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt); // Cálculo del ángulo utilizando filtro
228 Kalman
229
230 if (abs(kalAngleX) > 90)
231     gyroYrate = -gyroYrate; // Invierte la velocidad de los giróscopos, de forma que se restringe el
232 ángulo de los acelerómetros a -90/+90
233 kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
234
235 // Cálculo del ángulo mediante giróscopos sin ningún filtro
236 gyroXangle += gyroXrate * dt;
237 gyroYangle += gyroYrate * dt;
238
239 // Cálculo del ángulo usando un filtro complementario
240 compAngleX = 0.93 * (compAngleX + gyroXrate * dt) + 0.07 * roll;
241 compAngleY = 0.93 * (compAngleY + gyroYrate * dt) + 0.07 * pitch;
242
243 // Reseteo del ángulo de los giróscopos cuando se ha producido mucha deriva
244 if (gyroXangle < -180 || gyroXangle > 180)
245     gyroXangle = kalAngleX;
246 if (gyroYangle < -180 || gyroYangle > 180)
247     gyroYangle = kalAngleY;
248
249
250
251 //CÓDIGO PARA POSICIÓN, VELOCIDAD Y ACELERACIÓN //
252
253 if ((millis()-time1)>=50){
254     vel=(encoder0Pos-pos0)*1000/(millis()-time1);
255     pos0=encoder0Pos;
256     acel=(vel-vel0)/(millis()-time1);
257     velf=0.1*vel+(1-0.1)*vel0;
258     vel0=vel;
259     time1=millis();
260 }
261
262
263 if ((millis()-time2)>=50){
264     veld=(encoder1Pos-pos0d)*1000/(millis()-time2);
265     pos0d=encoder1Pos;
266     aceld=(veld-vel0d)/(millis()-time2);
267     velfd=0.1*veld+(1-0.1)*vel0d;
268     vel0d=veld;
269     time2=millis();
270 }
271
272 //CÓDIGO PARA CONTROLADOR //
273
274 lectpotP = analogRead(Pinpot1);
275 lectpotTi= analogRead(Pinpot2);
276 lectpotTd= analogRead(Pinpot3);
277
278 Kp=lectpotP/1023.0*40.0;
279 Ki=lectpotTi/1023.0*10;
280 Kd=lectpotTd/1023.0*2;
281
282 Ahora=millis();
283 Cambiot=Ahora-Ultimot;
284
285 if (Cambiot>=30){
286     e=r-kalAngleY;
287     ITerm += Ki*e;
288     if ((abs(ITerm))>255.0){
289         if (ITerm>0) ITerm=255.0;
290         else ITerm=-255.0;
291     }
292 }

```

```

293     dInput=kalAngleY-lastInput;
294     u=Kp*e+ITerm-Kd*dInput;
295
296     lastInput=kalAngleY;
297     Ultimot=Ahora;
298 }
299
300
301
302 if (u<=0){
303     digitalWrite(PinIn1I, HIGH);
304     digitalWrite(PinIn2I, LOW);
305     digitalWrite(PinIn1D, HIGH);
306     digitalWrite(PinIn2D, LOW);
307     PWMI=18+abs(u);
308     PWMD=16+abs(u);}
309 else {
310     digitalWrite(PinIn1I, LOW);
311     digitalWrite(PinIn2I, HIGH);
312     digitalWrite(PinIn1D, LOW);
313     digitalWrite(PinIn2D, HIGH);
314     PWMI=19+abs(u);
315     PWMD=17+abs(u);}
316
317 if (PWMI>255){
318     PWMI=255;
319     PWMD=PWMI;}
320 if (abs(e)>25){
321     PWMI=0;
322     PWMD=0;}
323 analogWrite(PinPWMI, PWMI);
324 analogWrite(PinPWMD, PWMD);
325
326 ///////////////////////////////////////////////////
327
328 /* Enviar datos */
329 #if 0 // Cambiar a 1 para activar
330     Serial.print(accX); Serial.print("\t");
331     Serial.print(accY); Serial.print("\t");
332     Serial.print(accZ); Serial.print("\t");
333
334     Serial.print(gyroX); Serial.print("\t");
335     Serial.print(gyroY); Serial.print("\t");
336     Serial.print(gyroZ); Serial.print("\t");
337
338     Serial.print("\t");
339 #endif
340
341 #if 0 // Cambiar a 1 para activar
342     Serial.print(roll); Serial.print("\t");
343     Serial.print(gyroXangle); Serial.print("\t");
344     Serial.print(compAngleX); Serial.print("\t");
345     Serial.print(kalAngleX); Serial.print("\t");
346
347     Serial.print("\t");
348
349     Serial.print(pitch); Serial.print("\t");
350     Serial.print(gyroYangle); Serial.print("\t");
351     Serial.print(compAngleY); Serial.print("\t");
352     Serial.print(kalAngleY); Serial.print("\t");
353 #endif
354
355 #if 0 // Cambiar a 1 para activar
356     Serial.print("\t");
357     double temperature = (double)tempRaw / 340.0 + 36.53; //Sensibilidad en Datasheet pag. 14
358     // (340LSB/°C) (offset=35°C segun del datasheet; pero es=36.53 en el pdf Mapa Registro)
359     Serial.print(temperature); Serial.print("\t");
360 #endif
361
362 #if 0
363     Serial.print("\t");
364
365     Serial.print(encoder0Pos); Serial.print("\t");
366     Serial.print(encoder1Pos); Serial.print("\t");
367     Serial.print(velf); Serial.print("\t");
368     Serial.print(velfd); Serial.print("\t");
369     Serial.print(accel); Serial.print("\t");
370     Serial.print(acceld); Serial.print("\t");
371
372     Serial.print("\t");
373
374     Serial.print(Kp); Serial.print("\t");
375     Serial.print(Ki); Serial.print("\t");
376     Serial.print(Kd); Serial.print("\t");
377
378     Serial.print("\r\n");
379     delay(2);
380
381 #endif
382 }
383
384 ////// A CONTINUACIÓN VIENEN LAS FUNCIONES DE LAS INTERRUPCIONES PARA LOS ENCODERS //////
385
386 //INTERRUPCIONES ENCODERS MOTOR IZQUIERDO
387 void doEncoderA0(){
388     if (digitalRead(encoder0PinA) == HIGH) { // busca un cambio de bajo-a-alto en el canal A
389         if (digitalRead(encoder0PinB) == LOW) { // comprueba el canal B para saber en qué sentido gira
390             encoder0Pos++;} // Horario
391         else {encoder0Pos--;} // Antihorario
392     }

```

```

393
394     else{ // en otro caso sería un cambio de alto-a-bajo en canal A
395         if (digitalRead(encoder0PinB) == HIGH) { // comprueba el canal B para saber en qué sentido gira
296             encoder0Pos++; // Horario
397         } else {encoder0Pos--;} // Antihorario
398     }
399 }
400
401 void doEncoderB0(){
402     if (digitalRead(encoder0PinB) == HIGH) { //busca un cambio de bajo-a-alto en el canal B
403         if (digitalRead(encoder0PinA) == HIGH) { // comprueba el canal A para saber en qué sentido gira
404             encoder0Pos++; // Horario
405         } else {encoder0Pos--;} // Antihorario
406     }
407
408     else { // en otro caso sería un cambio de alto-a-bajo en canal B
409         if (digitalRead(encoder0PinA) == LOW) { // comprueba el canal A para saber en qué sentido gira
410             encoder0Pos++; // Horario
411         } else {encoder0Pos--;} // Antihorario
412     }
413 }
414
415 //INTERRUPCIONES ENCODERS MOTOR DERECHO
416 void doEncoderA1(){
417     if (digitalRead(encoder1PinA) == HIGH) { // busca un cambio de bajo-a-alto en el canal A
418         if (digitalRead(encoder1PinB) == LOW) { // comprueba el canal B para saber en qué sentido gira
419             encoder1Pos--;} // Horario
420         } else {encoder1Pos++;} // Antihorario
421     }
422
423     else{ // en otro caso sería un cambio de alto-a-bajo en canal A
424         if (digitalRead(encoder1PinB) == HIGH) { // comprueba el canal B para saber en qué sentido gira
425             encoder1Pos--;} // Horario
426         } else {encoder1Pos++;} // Antihorario
427     }
428 }
429
430 void doEncoderB1(){
431     if (digitalRead(encoder1PinB) == HIGH) { //busca un cambio de bajo-a-alto en el canal B
432         if (digitalRead(encoder1PinA) == HIGH) { // comprueba el canal A para saber en qué sentido gira
433             encoder1Pos--;} // Horario
434         } else {encoder1Pos++;} // Antihorario
435     }
436
437     else { // en otro caso sería un cambio de alto-a-bajo en canal B
438         if (digitalRead(encoder1PinA) == LOW) { // comprueba el canal A para saber en qué sentido gira
439             encoder1Pos--;} // Horario
440         } else {encoder1Pos++;} // Antihorario
441     }
442 }
443 }
444
445
446
447
448
449
450
451
452
453

```

Archivo: CONTROLADOR_LQR.ino

```

1      #include <Wire.h>
2      #include "Kalman.h"
3      Kalman kalmanX;
4      Kalman kalmanY;
5
6      /* PARA LOS DATOS DE LA IMU */
7      double accX, accY, accZ;
8      double gyroX, gyroY, gyroZ;
9      int16_t tempRaw;
10     double gyroXangle, gyroYangle;
11     double compAngleX, compAngleY;
12     double kalAngleX, kalAngleY;
13
14     uint32_t timer;
15     uint8_t i2cData[14];
16     ///////////////////////////////////
17
18     /* PARA LOS ENCODERS, VELOCIDAD, FILTRADO Y ACELERACIÓN */
19     //Izquierdo
20     #define encoder0PinA 18
21     #define encoder0PinB 19
22     volatile float encoder0Pos = 0;
23     unsigned long time1;
24     float pos0;
25     float veli;
26
27     //Derecho
28     #define encoder1PinA 2
29     #define encoder1PinB 3
30     volatile float encoder1Pos = 0;
31     unsigned long time2;
32     float pos0d;
33     float veld;
34
35     //Filtro y acel
36     float wf, wf_1, wf_2, wf0;
37     float vel_1, vel_2, accel;

```

```

38 //////////////////////////////////////////////////
39
40 /* PARA EL MOTOR */
41 #define PinPWMD 7
42 #define PinPWMI 9
43 #define PinIn1I 40
44 #define PinIn2I 41
45 #define PinIn1D 43
46 #define PinIn2D 42
47 float PWMD;
48 float PWMI;
49 //////////////////////////////////////////////////
50
51 /* PARA EL CONTROL LQR*/
52 float u;
53 float phi_r,dphi_r,dtheta_r;
54 float phi, dphi;
55 float Ahora;
56 float Cambiot, Ultimot;
57 float Kp,Ki;
58 float r,e;
59 float Ahorai, Cambiot1, Ultimot1;
60 float ITerm;
61 float u_m;
62 //////////////////////////////////////////////////
63
64 /* PARA LAS MANIOBRAS DE PRUEBA */
65 float Tiempo0, Tiempo1;
66 //////////////////////////////////////////////////
67
68 void setup() {
69     Serial.begin(115200);
70
71     ////////// CÓDIGO IMU //////////
72     Wire.begin();
73     TWBR = ((F_CPU / 400000L) - 16) / 2;
74     i2cData[0] = 7;
75     i2cData[1] = 0x00;
76     i2cData[2] = 0x00;
77     i2cData[3] = 0x00;
78     while (i2cWrite(0x19, i2cData, 4, false));
79     while (i2cWrite(0x6B, 0x01, true));
80     while (i2cRead(0x75, i2cData, 1));
81     if (i2cData[0] != 0x68) {
82         Serial.print(F("Error leyendo sensor"));
83         while (1);
84     }
85     delay(100);
86
87     /* Ángulos iniciales cálculo kalman y con giróscopos*/
88     while (i2cRead(0x3B, i2cData, 6)
89     accX = (i2cData[0] << 8) | i2cData[1];
90     accY = (i2cData[2] << 8) | i2cData[3];
91     accZ = (i2cData[4] << 8) | i2cData[5];
92
93     // Posteriormente se convierte de radianes a grados
94     double roll =atan2(accY,accZ)*RAD_TO_DEG;
95     double pitch=atan(-accX/sqrt(accY*accY+accZ*accZ))*RAD_TO_DEG;
96
97     // Inicializar ángulos
98     kalmanX.setAngle(roll);
99     kalmanY.setAngle(pitch);
100     gyroXangle = roll;
101     gyroYangle = pitch;
102     compAngleX = roll;
103     compAngleY = pitch;
104     timer = micros();
105     //////////////////////////////////////
106
107     ////////// CÓDIGO PARA LAS INTERRUPCIONES //////////
108     //Izquierdo
109     pinMode(encoder0PinA, INPUT);
110     digitalWrite(encoder0PinA, HIGH);
111     pinMode(encoder0PinB, INPUT);
112     digitalWrite(encoder0PinB, HIGH);
113     attachInterrupt(5, doEncoderA0, CHANGE);
114     attachInterrupt(4, doEncoderB0, CHANGE);
115     //Derecho
116     pinMode(encoder1PinA, INPUT);
117     digitalWrite(encoder1PinA, HIGH);
118     pinMode(encoder1PinB, INPUT);
119     digitalWrite(encoder1PinB, HIGH);
120     attachInterrupt(0, doEncoderA1, CHANGE);
121     attachInterrupt(1, doEncoderB1, CHANGE);
122     time1=millis();
123     time2=millis();
124     //////////////////////////////////////
125
126     ////////// MOTOR //////////
127     TCCR4B = TCCR4B & 0b000 | 0x01;
128     TCCR2B = TCCR2B & 0b000 | 0x01;
129     pinMode(PinPWMD, OUTPUT);
130     pinMode(PinPWMI, OUTPUT);
131     pinMode(PinIn1I, OUTPUT);
132     pinMode(PinIn2I, OUTPUT);
133     pinMode(PinIn1D, OUTPUT);
134     pinMode(PinIn2D, OUTPUT);
135
136
137 //CONTROLADOR///

```

```

138 phi_r=0;dphi_r=0;dtheta_r=0;
139 u=0;
140 Kp=0.2;
141 Ki=0.1;
142 ///////////////////////////////////////////////////
143
144 ///FILTRO VELOCIDAD///
145 wf_2=1.0;
146 wf_1=1.0;
147 wf=1.0;
148 vel_2=1.0;
149 vel_1=1.0;
150 wf0=1.0;
151 ///////////////////////////////////////////////////
152
153 //CONFIGURACIÓN PARA MANIOBRAS//
154 Tiempo0=millis();
155 ///////////////////////////////////////////////////
156 }
157
158
159 void loop() {
160
161     ///CÓDIGO PARA LA IMU///
162     /* Actualización de todos los valores */
163     while (i2cRead(0x3B, i2cData, 14));
164     accX = ((i2cData[0] << 8) | i2cData[1]) - 432.0;
165     accY = ((i2cData[2] << 8) | i2cData[3]) - 291.16;
166     accZ = ((i2cData[4] << 8) | i2cData[5]) - 307.52;
167     tempRaw = (i2cData[6] << 8) | i2cData[7];
168     gyroX = (i2cData[8] << 8) | i2cData[9];
169     gyroY = (i2cData[10] << 8) | i2cData[11];
170     gyroZ = (i2cData[12] << 8) | i2cData[13];
171     double dt = (double)(micros()- timer) / 1000000; // Cálculo de dt
172     timer = micros();
173
174     // atan2 devuelve valores de -π a π (radianes)
175     // Posteriormente se convierte de radianes a grados
176     double roll = atan2(accY, accZ) * RAD_TO_DEG;
177     double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
178     double gyroXrate = gyroX / 131.0;
179     double gyroYrate = gyroY / 131.0;
180
181     if ((roll < -90 && kalAngleX > 90) || (roll > 90 && kalAngleX < -90)) {
182         kalmanX.setAngle(roll);
183         compAngleX = roll;
184         kalAngleX = roll;
185         gyroXangle = roll;
186     } else
187         kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);
188
189     if (abs(kalAngleX) > 90)
190         gyroYrate = -gyroYrate;
191     kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
192
193     // Cálculo del ángulo mediante giróscopos sin ningún filtro
194     gyroXangle += gyroXrate * dt;
195     gyroYangle += gyroYrate * dt;
196     // Cálculo del ángulo usando un filtro complementario
197     compAngleX = 0.93 * (compAngleX + gyroXrate * dt) + 0.07 * roll;
198     compAngleY = 0.93 * (compAngleY + gyroYrate * dt) + 0.07 * pitch;
199     // Reseteo del ángulo de los giróscopos cuando se ha producido mucha deriva
200     if (gyroXangle < -180 || gyroXangle > 180)
201         gyroXangle = kalAngleX;
202     if (gyroYangle < -180 || gyroYangle > 180)
203         gyroYangle = kalAngleY;
204     ///////////////////////////////////
205
206
207
208
209     ////////// CÓDIGO PARA POSICIÓN, VELOCIDAD Y ACELERACIÓN //////////
210     if ((millis()-time1)>=10){
211         veli=((encoder0Pos-pos0)*1000.0/(millis()-time1))*(PI/180.0);
212         pos0=encoder0Pos;
213         veld=((encoder1Pos-pos0d)*1000.0/(millis()-time1))*(PI/180.0);
214         pos0d=encoder1Pos;
215         vel=(veli+veld)/2;
216
217         //Filtrado de la velocidad(BUTTERWORTH CON N=2; Wn=0.0784):
218         wf=1.6544*wf_1 - 0.7059*wf_2 + 0.0129*vel + 0.0257*vel_1 + 0.0129*vel_2;
219         wf_2=wf_1;
220         wf_1=wf;
221         vel_2=vel_1;
222         vel_1=vel;
223
224         acelf=(wf-wf0)*1000.0/(millis()-time1);
225         wf0=wf;
226         time1=millis();}
227     ///////////////////////////////////
228
229
230     //////////CÓDIGO PARA MANIOBRAS DEL ROBOT //////////
231     Tiempo1=millis();
232     if ((Tiempo1-Tiempo0)<4000){
233         dtheta_r=0;}
234     else {if ((Tiempo1-Tiempo0)<9000){
235         dtheta_r=3.8*PI;}
236         else{ if ((Tiempo1-Tiempo0)<13000){
237             dtheta_r=0;}

```

```

238         else{if ((Tiempo1-Tiempo0)<18000){
239             dtheta_r=-3.8*PI;}
240         else{dtheta_r=0;}
241     }
242 }
243 }
244 //////////////////////////////////////////////////
245
246
247 ////////////////////////////////////////////////// CÓDIGO PARA CONTROLADOR //////////////////////////////////
248 Ahora=millis();
249 Cambiot=Ahora-Ultimot;
250
251 if (Cambiot>=40){
252     phi=kAngleY*(PI/180);
253     dphi=gyroYrate*(PI/180);
254     u=-332.8746*(phi_r-phi)-52.7835*(dphi_r-dphi)-3.1623*(dtheta_r-wf);
255     r=-u*1.9;
256     Ultimot=Ahora;}
257 Ahorat=millis();
258 Cambiot1=Ahorat-Ultimot1;
259
260 if (Cambiot1>=15){
261     Ultimot1=Ahorat;
262     e=r-acelf;
263     ITerm += Ki*e;
264     if((abs(ITerm))>255.0){
265         if (ITerm>0) ITerm=255.0;
266         else ITerm=-255.0;}
267     u_m=Kp*e+ITerm;}
268
269
270 if (u_m<=0){
271     digitalWrite(PinIn1I, HIGH);
272     digitalWrite(PinIn2I, LOW);
273     digitalWrite(PinIn1D, HIGH);
274     digitalWrite(PinIn2D, LOW);
275     PWMI=17+abs(u_m);
276     PWMD=16+abs(u_m);}
277 else {
278     digitalWrite(PinIn1I, LOW);
279     digitalWrite(PinIn2I, HIGH);
280     digitalWrite(PinIn1D, LOW);
281     digitalWrite(PinIn2D, HIGH);
282     PWMI=18+abs(u_m);
283     PWMD=17+abs(u_m);}
284
285 if (PWMI>255){
286     PWMI=255;
287     PWMD=PWMI;}
288 if (abs(kAngleY)>30){
289     PWMI=0;
290     PWMD=0;}
291 analogWrite(PinPWMI, PWMI);
292 analogWrite(PinPWMD, PWMD);
293 //////////////////////////////////////////////////
294
295
296 /* ENVÍO DE DATOS */
297 #if 0 // Cambiar a 1 para activar
298     Serial.print(r); Serial.print("\t");
299     Serial.print(acel); Serial.print("\t");
300     Serial.print(phi); Serial.print("\t");
301     Serial.print(dphi); Serial.print("\t");
302     Serial.print(vel); Serial.print("\t");
303     Serial.print(millis()); Serial.print("\t");
304     Serial.print("\t");
305 #endif
306 #if 0 // Cambiar a 1 para activar
307     Serial.print(accX); Serial.print("\t");
308     Serial.print(accY); Serial.print("\t");
309     Serial.print(accZ); Serial.print("\t");
310
311     Serial.print(gyroX); Serial.print("\t");
312     Serial.print(gyroY); Serial.print("\t");
313     Serial.print(gyroZ); Serial.print("\t");
314     Serial.print("\t");
315 #endif
316 #if 0 // Cambiar a 1 para activar
317     Serial.print(roll); Serial.print("\t");
318     Serial.print(gyroXangle); Serial.print("\t");
319     Serial.print(compAngleX); Serial.print("\t");
320     Serial.print(kalAngleX); Serial.print("\t");
321     Serial.print("\t");
322     Serial.print(pitch); Serial.print("\t");
323     Serial.print(gyroYangle); Serial.print("\t");
324     Serial.print(compAngleY); Serial.print("\t");
325     Serial.print(kalAngleY); Serial.print("\t");
326 #endif
327 #if 0 // Cambiar a 1 para activar
328     Serial.print("\t");
329     double temperature = (double)tempRaw / 340.0 + 36.53;
330     Serial.print(temperature); Serial.print("\t");
331 #endif
332 #if 0
333     Serial.print("\t");
334     Serial.print(encoder0Pos); Serial.print("\t");
335     Serial.print(encoder1Pos); Serial.print("\t");
336     Serial.print(velf); Serial.print("\t");
337     Serial.print(velfd); Serial.print("\t");

```



```

338     Serial.print(accel); Serial.print("\t");
339     Serial.print(acceld); Serial.print("\t");
340     Serial.print("\t");
341     Serial.print(Kp); Serial.print("\t");
342     Serial.print(Ki); Serial.print("\t");
343     Serial.print(Kd); Serial.print("\t");
344 #endif
345 #if 0 // Cambiar a 1 para activar
346     Serial.print(u); Serial.print("\t");
347     Serial.print(FWMI); Serial.print("\t");
348     Serial.print(kalAngleY); Serial.print("\t");
349 #endif
350     Serial.print("\r\n");
351     delay(1);
352     //////////////////////////////////////
353 }
354
355 // A CONTINUACIÓN VIENEN LAS FUNCIONES DE LAS
356 // INTERRUPTIONES PARA LOS ENCODERS //////////////////////////////////
357 // INTERRUPTIONES ENCODERS MOTOR IZQUIERDO
358 void doEncoderA0() {
359     if (digitalRead(encoder0PinA) == HIGH) {
360         if (digitalRead(encoder0PinB) == LOW) {
361             encoder0Pos++; // Horario
362         } else {encoder0Pos--;} // Antihorario
363     }
364     else {
365         if (digitalRead(encoder0PinB) == HIGH) {
366             encoder0Pos++; // Horario
367         } else {encoder0Pos--;} // Antihorario
368     }
369 }
370 void doEncoderB0() {
371     if (digitalRead(encoder0PinB) == HIGH) {
372         if (digitalRead(encoder0PinA) == HIGH) {
373             encoder0Pos++; // Horario
374         } else {encoder0Pos--;} // Antihorario
375     }
376     else {
377         if (digitalRead(encoder0PinA) == LOW) {
378             encoder0Pos++; // Horario
379         } else {encoder0Pos--;} // Antihorario
380     }
381 }
382
383 // INTERRUPTIONES ENCODERS MOTOR DERECHO
384 void doEncoderA1() {
385     if (digitalRead(encoder1PinA) == HIGH) {
386         if (digitalRead(encoder1PinB) == LOW) {
387             encoder1Pos--; // Horario
388         } else {encoder1Pos++;} // Antihorario
389     }
390     else {
391         if (digitalRead(encoder1PinB) == HIGH) {
392             encoder1Pos--;} // Horario
393         } else {encoder1Pos++;} // Antihorario
394     }
395 }
396 void doEncoderB1() {
397     if (digitalRead(encoder1PinB) == HIGH) {
398         if (digitalRead(encoder1PinA) == HIGH) {
399             gira
400                 encoder1Pos--;} // Horario
401             } else {encoder1Pos++;} // Antihorario
402         }
403     }
404     else {
405         if (digitalRead(encoder1PinA) == LOW) {
406             encoder1Pos--;} // Horario
407         } else {encoder1Pos++;} // Antihorario
408     }
409 }
410

```

Archivo: CONTROLADOR_LQR_CON_BLUETOOTH.ino

```

1  #include <Wire.h>
2  #include "Kalman.h"
3  Kalman kalmanX;
4  Kalman kalmanY;
5
6  /* PARA LOS DATOS DE LA IMU */
7  double accX, accY, accZ;
8  double gyroX, gyroY, gyroZ;
9  int16_t tempRaw;
10 double gyroXangle, gyroYangle;
11 double compAngleX, compAngleY;
12 double kalAngleX, kalAngleY;
13
14 uint32_t timer;
15 uint8_t i2cData[14];

```

```

16 //////////////////////////////////////////////////
17
18 /* PARA LOS ENCODERS, VELOCIDAD, FILTRADO Y ACELERACIÓN */
19 //Izquierdo
20 #define encoderOPinA 18
21 #define encoderOPinB 19
22 volatile float encoderOPos = 0;
23 unsigned long time1;
24 float pos0;
25 float veli;
26
27 //Derecho
28 #define encoder1PinA 2
29 #define encoder1PinB 3
30 volatile float encoder1Pos = 0;
31 unsigned long time2;
32 float pos0d;
33 float veld;
34
35 //Filtro y acel
36 float wf, wf_1, wf_2, wf0;
37 float vel_1, vel_2, acelf;
38 //////////////////////////////////////////////////
39
40 /* PARA EL MOTOR */
41 #define PinPWMD 7
42 #define PinPWMI 9
43 #define PinIn1I 40
44 #define PinIn2I 41
45 #define PinIn1D 43
46 #define PinIn2D 42
47 float PWMD;
48 float PWMI;
49 //////////////////////////////////////////////////
50
51 /* PARA EL CONTROL LQR*/
52 float u;
53 float phi_r, dphi_r, dtheta_r;
54 float phi, dphi;
55 float Ahora;
56 float Cambiot, Ultimot;
57 float Kp,Ki;
58 float r,e;
59 float Ahorai, Cambiot1, Ultimot1;
60 float ITerm;
61 float u_m;
62 //////////////////////////////////////////////////
63
64 /* PARA MANIOBRAS*/
65
66 float u_mi, u_md;
67 float tiempoDir;
68 float Estado;
69
70 /* PARA EL FILTRO DE LA VELOCIDAD*/
71
72 float wf, wf_1, wf_2, wf0;
73 float vel_1, vel_2, acelf;
74 float veli, veld;
75
76
77 /* PARA LAS VAR DE LA COMUNICACIÓN BLUETOOTH*/
78 float K1, K2, K3, c; //variables para las constantes del LQR
79 int Dir; //variable para la dirección (de 1 a 9)
80 float xg, xt;
81 int trim;
82
83 float K1_r, K2_r, K3_r, c_r; //variables para las constantes en proceso de recepción del LQR
84 int Dir_r; //variable para la dirección (de 1 a 9)
85 float xg_r, xt_r;
86 float trim_r;
87
88 float Kp_r, Ki_r; //para la recepcion constantes control motor
89
90 float tiempo0; //Para el cálculo de tiempo de bucle de programa.
91 float bucle;
92
93 /* PARA EL TRIMADO*/
94 float trim_estado, adiccion, trimado;
95 float kalAngleY trimado;
96 //////////////////////////////////////////////////
97
98 void setup() {
99     Serial.begin(115200);
100
101     ////////////////////////////////////////////////// CÓDIGO IMU //////////////////////////////////////////////////
102     Wire.begin();
103     TWBR = ((F_CPU / 400000L) - 16) / 2;
104     i2cData[0] = 7;
105     i2cData[1] = 0x00;
106     i2cData[2] = 0x00;
107     i2cData[3] = 0x00;
108     while (i2cWrite(0x19, i2cData, 4, false));
109     while (i2cWrite(0x6B, 0x01, true));
110     while (i2cRead(0x75, i2cData, 1));
111     if (i2cData[0] != 0x68) {
112         Serial.print(F("Error leyendo sensor"));
113         while (1);
114     }
115     delay(100);

```

```

116
117 /* Ángulos iniciales cálculo kalman y con giróscopos*/
118 while (i2cRead(0x3B, i2cData, 6)
119 accX = (i2cData[0] << 8) | i2cData[1];
120 accY = (i2cData[2] << 8) | i2cData[3];
121 accZ = (i2cData[4] << 8) | i2cData[5];
122
123 // Posteriormente se convierte de radianes a grados
124 double roll =atan2(accY,accZ)*RAD_TO_DEG;
125 double pitch=atan(-accX/sqrt(accY*accY+accZ*accZ))*RAD_TO_DEG;
126
127 // Inicializar ángulos
128 kalmanX.setAngle(roll);
129 kalmanY.setAngle(pitch);
130 gyroXangle = roll;
131 gyroYangle = pitch;
132 compAngleX = roll;
133 compAngleY = pitch;
134 timer = micros();
135 ///////////////////////////////////////////////////
136
137
138 /////// CÓDIGO PARA LAS INTERRUPCIONES /////
139 //Izquierdo
140 pinMode(encoder0PinA, INPUT);
141 digitalWrite(encoder0PinA, HIGH);
142 pinMode(encoder0PinB, INPUT);
143 digitalWrite(encoder0PinB, HIGH);
144 attachInterrupt(5, doEncoderA0, CHANGE);
145 attachInterrupt(4, doEncoderB0, CHANGE);
146 //Derecho
147 pinMode(encoder1PinA, INPUT);
148 digitalWrite(encoder1PinA, HIGH);
149 pinMode(encoder1PinB, INPUT);
150 digitalWrite(encoder1PinB, HIGH);
151 attachInterrupt(0, doEncoderA1, CHANGE);
152 attachInterrupt(1, doEncoderB1, CHANGE);
153 time1=millis();
154 time2=millis();
155 ///////////////////////////////////////////////////
156
157 ////////// MOTOR ///////////
158 TCCR4B = TCCR4B & 0b000 | 0x01;
159 TCCR2B = TCCR2B & 0b000 | 0x01;
160 pinMode(PinPWMD, OUTPUT);
161 pinMode(PinPWWI, OUTPUT);
162 pinMode(PinIn1I, OUTPUT);
163 pinMode(PinIn2I, OUTPUT);
164 pinMode(PinIn1D, OUTPUT);
165 pinMode(PinIn2D, OUTPUT);
166
167 //CONTROLADOR///
168 phi_r=0;dphi_r=0;dtheta_r=0;
169 u=0;
170 Kp=0.2;//Kp=0.1;
171 Ki=0.1;//Ki=0.4;
172 K1=-332.8746; K2=-52.7835; K3=-3.1623; c=1.9; Dir=5.0; xg=1.0; xt=1.0;
173 ///////////////////////////////////////////////////
174
175
176 //FILTRO VELOCIDAD///
177 wf_2=1.0;
178 wf_1=1.0;
179 wf=1.0;
180 vel_2=1.0;
181 vel_1=1.0;
182 wf0=1.0;
183 ///////////////////////////////////////////////////
184
185 }
186
187
188 void loop() {
189
190 RecepcionBT();
191
192 //////CÓDIGO PARA LA IMU/////
193 /* Actualización de todos los valores */
194 while (i2cRead(0x3B, i2cData, 14));
195 accX = ((i2cData[0] << 8) | i2cData[1]) - 432.0;
196 accY = ((i2cData[2] << 8) | i2cData[3]) - 291.16;
197 accZ = ((i2cData[4] << 8) | i2cData[5]) - 307.52;
198 tempRaw = (i2cData[6] << 8) | i2cData[7];
199 gyroX = (i2cData[8] << 8) | i2cData[9];
200 gyroY = (i2cData[10] << 8) | i2cData[11];
201 gyroZ = (i2cData[12] << 8) | i2cData[13];
202 double dt = (double)(micros()- timer) / 1000000; // Cálculo de dt
203 timer = micros();
204
205 // atan2 devuelve valores de -pi a pi (radianes)
206 // Posteriormente se convierte de radianes a grados
207 double roll = atan2(accY, accZ) * RAD_TO_DEG;
208 double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
209 double gyroXrate = gyroX / 131.0;
210 double gyroYrate = gyroY / 131.0;
211
212 if ((roll < -90 && kalAngleX > 90) || (roll > 90 && kalAngleX < -90)) {
213 kalmanX.setAngle(roll);
214 compAngleX = roll;
215 kalAngleX = roll;

```

```

216     gyroXangle = roll;
217 } else
218     kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);
219
220 if (abs(kalAngleX) > 90)
221     gyroYrate = -gyroYrate;
222 kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
223
224 // Cálculo del ángulo mediante giróscopos sin ningún filtro
225 gyroXangle += gyroXrate * dt;
226 gyroYangle += gyroYrate * dt;
227 // Cálculo del ángulo usando un filtro complementario
228 compAngleX = 0.93 * (compAngleX + gyroXrate * dt) + 0.07 * roll;
229 compAngleY = 0.93 * (compAngleY + gyroYrate * dt) + 0.07 * pitch;
230 // Reseteo del ángulo de los giróscopos cuando se ha producido mucha deriva
231 if (gyroXangle < -180 || gyroXangle > 180)
232     gyroXangle = kalAngleX;
233 if (gyroYangle < -180 || gyroYangle > 180)
234     gyroYangle = kalAngleY;
235 ///////////////////////////////////////////////////
236
237
238
239
240 /////////////////////////////////////////////////// CÓDIGO PARA POSICIÓN, VELOCIDAD Y ACELERACIÓN ///////////////////////////////////////////////////
241 if ((millis() - time1) >= 10) {
242     veli = ((encoder0Pos - pos0) * 1000.0 / (millis() - time1)) * (PI / 180.0);
243     pos0 = encoder0Pos;
244     veld = ((encoder1Pos - pos0d) * 1000.0 / (millis() - time1)) * (PI / 180.0);
245     pos0d = encoder1Pos;
246     vel = (veli + veld) / 2;
247
248     //Filtrado de la velocidad (BUTTERWORTH CON N=2; Wn=0.0784):
249     wf = 1.6544 * wf_1 - 0.7059 * wf_2 + 0.0129 * vel + 0.0257 * vel_1 + 0.0129 * vel_2;
250     wf_2 = wf_1;
251     wf_1 = wf;
252     vel_2 = vel_1;
253     vel_1 = vel;
254
255     acelf = (wf - wf0) * 1000.0 / (millis() - time1);
256     wf0 = wf;
257     time1 = millis();
258     ///////////////////////////////////////////////////
259
260
261 /////////////////////////////////////////////////// CÓDIGO PARA CONTROLADOR ///////////////////////////////////////////////////
262
263 Ahora = millis();
264 Cambiot = Ahora - Ultimot;
265 Trimado();
266
267 if (Cambiot >= 40) {
268     phi = kalAngleY_trimado * (PI / 180);
269     dphi = gyroYrate * (PI / 180);
270     u = K1 * (phi_r - phi) + K2 * (dphi_r - dphi) + K3 * (dtheta_r - wf);
271     r = u * c; //Voy a tocar esto a ver si responde más rápido.
272     Ultimot = Ahora;
273 }
274
275 Ahorai = millis();
276 Cambiot1 = Ahorai - Ultimot1;
277
278 if (Cambiot1 >= 15) {
279     Ultimot1 = Ahorai;
280     e = r - acelf;
281     ITerm += Ki * e;
282     if ((abs(ITerm)) > 255.0) {
283         if (ITerm > 0) ITerm = 255.0;
284         else ITerm = -255.0;
285     }
286     u_m = Kp * e + ITerm;
287 }
288
289 Mando_Motor();
290
291 ///////////////////////////////////////////////////
292
293 /*SECCIÓN PARA EL ENVÍO*/
294
295 bucle = millis() - tiempo0;
296 tiempo0 = millis();
297
298 EnvioBT();
299 ///////////////////////////////////////////////////
300 }
301
302 // A CONTINUACIÓN VIENEN LAS FUNCIONES DE LAS
303 // INTERRUPTIONES PARA LOS ENCODERS ///////////////////////////////////////////////////
304 // INTERRUPTIONES ENCODERS MOTOR IZQUIERDO
305 void doEncoderA0() {
306     if (digitalRead(encoder0PinA) == HIGH) {
307         if (digitalRead(encoder0PinB) == LOW)
308             encoder0Pos++; // Horario
309         else {encoder0Pos--;} // Antihorario
310     }
311     else {
312         if (digitalRead(encoder0PinB) == HIGH) {
313             encoder0Pos++; // Horario
314         }
315         else {encoder0Pos--;} // Antihorario
316     }
317 }

```

```

316     }
317     void doEncoderB0(){
318         if (digitalRead(encoder0PinB) == HIGH) {
319             if (digitalRead(encoder0PinA) == HIGH) {
320                 encoder0Pos++;} // Horario
321             else {encoder0Pos--;} // Antihorario
322         }
323         else {
324             if (digitalRead(encoder0PinA) == LOW) {
325                 encoder0Pos++;} // Horario
326             else {encoder0Pos--;} // Antihorario
327         }
328     }
329
330     //INTERRUPCIONES ENCODERS MOTOR DERECHO
331     void doEncoderA1(){
332         if (digitalRead(encoder1PinA) == HIGH) {
333             if (digitalRead(encoder1PinB) == LOW) {
334                 encoder1Pos--;} // Horario
335             else {encoder1Pos++;} // Antihorario
336         }
337         else{
338             if (digitalRead(encoder1PinB) == HIGH) {
339                 encoder1Pos--;} // Horario
340             else {encoder1Pos++;} // Antihorario
341         }
342     }
343     void doEncoderB1(){
344         if (digitalRead(encoder1PinB) == HIGH) {
345             if (digitalRead(encoder1PinA) == HIGH) {
346                 gira
347                     encoder1Pos--;} // Horario
348                 else {encoder1Pos++;} // Antihorario
349             }
350
351             else {
352                 if (digitalRead(encoder1PinA) == LOW) {
353                     encoder1Pos--;} // Horario
354                 else {encoder1Pos++;} // Antihorario
355             }
356         }
357     }
358     //////////////////////////////////////
359     //// A CONTINUACIÓN VIENE EL RESTO LAS FUNCIONES UTILIZADAS
360
361     void Mando_Motor(){
362         switch(Dir){
363             case 1:
364                 if ((millis()-tiempoDir)<1000){
365                     u_md=-u_m+10.0*xg;
366                     u_mi=-u_m-10.0*xg;
367                     Estado=10;}
368                 if ((millis()-tiempoDir)>=1000){
369                     dtheta_r=0.76*PI*xt;
370                     u_mi=-u_m;
371                     u_md=u_mi;
372                     Estado=101;}
373                 break;
374
375             case 2:
376                 dtheta_r=0.76*PI*xt;
377                 u_mi=-u_m;
378                 u_md=u_mi;
379                 Estado=20;
380                 break;
381
382             case 3:
383                 if ((millis()-tiempoDir)<1000){
384                     u_mi=-u_m+10.0*xg;
385                     u_md=-u_m-10.0*xg;
386                     Estado=30;}
387                 if ((millis()-tiempoDir)>=1000){
388                     dtheta_r=0.76*PI*xt;
389                     u_mi=-u_m;
390                     u_md=u_mi;
391                     Estado=301;}
392                 break;
393
394             case 4:
395                 u_md=-u_m+10.0*xg;
396                 u_mi=-u_m-10.0*xg;
397                 Estado=40;
398                 break;
399
400             case 5:
401                 dtheta_r=0;
402                 u_mi=-u_m;
403                 u_md=u_mi;
404                 Estado=50;
405                 break;
406
407             case 6:
408                 u_mi=-u_m+10.0*xg;
409                 u_md=-u_m-10.0*xg;
410                 Estado=60;
411                 break;
412
413             case 7:
414                 if ((millis()-tiempoDir)<1000){
415                     u_mi=-u_m+10.0*xg;

```

```

416         u_md=-u_m-10.0*xg;
417         Estado=70;}
418     if ((millis()-tiempoDir)>=1000){
419         dtheta_r=-0.76*PI*xt;
420         u_mi=-u_m;
421         u_md=u_mi;
422         Estado=701;}
423     break;
424
425     case 8:
426         dtheta_r=-0.76*PI*xt;
427         u_mi=-u_m;
428         u_md=u_mi;
429         Estado=80;
430     break;
431
432     case 9:
433     if ((millis()-tiempoDir)<1000){
434         u_md=-u_m+10.0*xg;
435         u_mi=-u_m-10.0*xg;
436         Estado=90;}
437     if ((millis()-tiempoDir)>=1000){
438         dtheta_r=-0.76*PI*xt;
439         u_mi=-u_m;
440         u_md=u_mi;
441         Estado=901;}
442     break;
443
444     default:
445         dtheta_r=0;
446         u_mi=-u_m;
447         u_md=u_mi;
448         Estado=112;
449 }
450
451 if (u_mi>=0){
452     digitalWrite(PinIn1I, HIGH);
453     digitalWrite(PinIn2I, LOW);
454     PWMI=17+abs(u_mi);}
455 else {
456     digitalWrite(PinIn1I, LOW);
457     digitalWrite(PinIn2I, HIGH);
458     PWMI=18+abs(u_mi);}
459
460
461 if (u_md>=0){
462     digitalWrite(PinIn1D, HIGH);
463     digitalWrite(PinIn2D, LOW);
464     PWMD=16+abs(u_md);}
465 else {
466     digitalWrite(PinIn1D, LOW);
467     digitalWrite(PinIn2D, HIGH);
468     PWMD=17+abs(u_md);}
469
470
471 if (PWMI>255){
472     PWMI=255;}
473 if (PWMD>255){
474     PWMD=255;}
475
476 if (abs(kalAngleY)>30){
477     PWMI=0;
478     PWMD=0;}
479
480 analogWrite(PinPWMI, PWMI);
481 analogWrite(PinPWMD, PWMD);
482
483 }
484 ///////////////////////////////////////////////////
485
486
487
488 void RecepcionBT(){
489     while (Serial.available()>0)
490     {
491         xg_r = Serial.parseFloat(); //Esta es la velocidad de giro, la posición del slider.
492         xt_r = Serial.parseFloat();
493         trim_r = Serial.parseInt(); //Esta es la velocidad de traslacion, la posición del slider.
494         Kp_r = Serial.parseFloat();
495         Ki_r = Serial.parseFloat();
496         K1_r = Serial.parseFloat();
497         K2_r = Serial.parseFloat();
498         K3_r = Serial.parseFloat();
499         c_r = Serial.parseFloat();
500         Dir_r = Serial.parseInt();
501
502         //Cuando lea el carácter fin de línea ('\n') quiere decir que ha finalizado el envío
503         if (Serial.read() == '\n')
504         {
505             //Actualizamos las variables reales si procede
506             if (K1_r != 0)
507                 {Kp=Kp_r; Ki=Ki_r; K1=K1_r; K2=K2_r; K3=K3_r; c=c_r;}
508
509             if (Dir_r != 0)
510                 {Dir=Dir_r;
511                  if (Dir==1||Dir==3||Dir==7||Dir==9)
512                      {tiempoDir=millis();}
513                  }
514
515             if (xg_r != 0)
516                 {xg=xg_r;}

```

```

467
468         if (xt_r != 0)
469             {xt=xt_r;}
470
471         if (trim_r != 0)
472             {trim=trim_r;
473              trim_estado=1;
474             }
475
476     }
477
478 }
479
480 }
481
482 void EnvioBT() {
483     Serial.print("^");
484     Serial.print(Kp); Serial.print(",");
485     Serial.print(Ki); Serial.print(",");
486     Serial.print(K1); Serial.print(",");
487     Serial.print(K2); Serial.print(",");
488     Serial.print(K3); Serial.print(",");
489     Serial.print(c); Serial.print(",");
490     Serial.print(bucle); Serial.print(","); //Tiempo que tarda en dar una vuelta el programa.
491     Serial.print(Dir); Serial.print(",");
492     Serial.print(Estado); Serial.print(",");
493     Serial.print(xg); Serial.print(",");
494     Serial.print(xt); Serial.print(",");
495     Serial.print(trimado);
496     delay(1);
497 }
498
499 //////////////////////////////////////
500
501 void Trimado() {
502     if (trim_estado==1) {
503         switch(trim) {
504             case 1:
505                 adiccion = -0.5; //Cada vez que pulsamos uno de los botones de trimar en la aplicación android subimos o bajamos medio grado.
506                 break;
507
508             case 2:
509                 adiccion = 0.5;
510                 break;
511
512             default:
513                 adiccion=0;
514         }
515         trim_estado=0;
516     }
517     else {adiccion=0;}
518     trimado+=adiccion;
519     kalAngleY_trimado= kalAngleY+trimado;
520 }

```