APÉNDICE D: Código en *Arduino*

Archivo: CONTROLADOR_PID.ino

```
#include <Wire.h>   // Esta librería permite  comunicar con dispositivos I2C  TWI.
#include "Kalman.h" // Carga de la libreria Filtro Kalman

Kalman kalmanX; //
Kalman kalmanY; //Aquí está creando los objetos kalmanX y KalmanY de tipo Kalman(librería). Más
adelante si quiero usar una función dentro de Kalman.h la usaré particularizada al objeto haciendo:
kalmanX.funcion()o kalmanY.funcion()

/* Datos de IMU */
double accX, accY, accZ;
double gyroX, gyroY, gyroZ;
int16_t tempRaw;  //Aquí se almacenará la temperatura en crudo (no en °C)

double gyroXangle, gyroYangle; // Ángulo calculado utilizando sólo los giróscopos
double compAngleX, compAngleY; // Ángulo calculado utilizando un filtro complementario
double kalAngleX, kalAngleY;   // Ángulo calculado utilizando un filtro Kalman

uint32_t timer;
uint8_t i2cData[14]; // Buffer para datos I2C. Creación del vector i2cData reservando 14 elementos


/* PARA LOS ENCODERS, VELOCIDAD Y ACELERACIÓN */
//Izquierdo
#define encoder0PinA 18
#define encoder0PinB 19
volatile float encoder0Pos = 0;
unsigned long time1;
float pos0;
float vel;
float vel0;
float acel;
float velf;

//Derecho
#define encoder1PinA 2
#define encoder1PinB 3
volatile float encoder1Pos = 0;
unsigned long time2;
float pos0d;
float veld;
float vel0d;
float aceld;
float velfd;

//////////////////////////

/* PARA EL MOTOR */

#define PinPWMD 7
#define PinPWMI 9
#define PinIn1I 40
#define PinIn2I 41
#define PinIn1D 43
#define PinIn2D 42

float PWMD;
float PWMI;

//////////////////////////

/* PARA EL CONTROL PID*/

#define Pinpot1 A2
#define Pinpot2 A1
#define Pinpot3 A0
int lectpotP;
int lectpotTi;
int lectpotTd;

float u;
float r;
float e;
float Kp;
float Ki;
float Kd;
float dInput, lastInput;
float ITerm;
float Ahora;
float Cambiot, Ultimot;

//////////////////////////

void setup() {
  Serial.begin(115200);
  Wire.begin();
  TWBR = ((F_CPU / 400000L) - 16) / 2; // Configurar la frecuencia I2C a 400kHz  TWBR - TWI Bit Rate
Register --> He econtrado esto: TWBR selects the division factor for the bit rate generator. The bit
rate generator is a frequency divider which generates the SCL clock frequency in the Master modes.
                                       //En el Datasheet de la MPU6050 dicen que la máx. velocidad
del bus es 400kHz (por eso él lo pone a esta vel.)

  i2cData[0] = 7;    // Configurar la tasa de muestreo a 1000Hz -> 8kHz/(7+1) = 1000Hz
```

```
  >>Config tasa muestreo.        Registro 0x19hex=25dec (Nuestro giroscopo funciona a 8kHz, ver pag.12-13 Mapa Registro pdf)
    i2cData[1] = 0x00; // Deshabilitar FSYNC y configurar los filtros a 260 Hz Acel., 256 Hz Giros., y muestreo 8 KHz  >>Sincronización y Filtrado.   Registro 0x1Ahex=26dec (ver pag.13 Mapa Registro pdf)
    i2cData[2] = 0x00; // Configurar Rango Giroscopos a ±250deg/s
  >>Configuración Giróscopos.    Registro 0x1Bhex=27dec Lo pone a máxima sensibilidad/minimo rango y deshabilita el autotest(ver pag.14 Mapa Registro pdf)
    i2cData[3] = 0x00; // Configurar Rango Acelerometros a ±2g
  >>Configuración Acelerómetros. Registro 0x1Chex=28dec Lo pone a máxima sensibilidad/minimo rango y deshabilita el autotest (ver pag.15-16 Mapa Registro pdf)
    while (i2cWrite(0x19, i2cData, 4, false)); //  >>Meter los anteriores 4 valores en cada posición del registro y es un while porque devuelve un 0 cuando es con éxito y si no lo reintenta (ver archivo i2C.ino)
    while (i2cWrite(0x6B, 0x01, true)); //         >>Deshabilitar modo standby. Usar como señal de reloj el giroscopo del eje X. Esto se mete en el registro 0x6Bhex=107dec (ver pag.41 Mapa Registro pdf)

    while (i2cRead(0x75, i2cData, 1));  //Leer los registros 0x75hex=117dec; y los mete en el vector i2cData. Como el registro 0x75 es el último, el resto de valores del vector quedará con los valores que tenía antes.
    if (i2cData[0] != 0x68) { // Leer el registro "Quién Soy?" (ver pag. 46) devuelve el valor 0x68 para decir cuál es la dirección del dispositivo (aunque realmente sabemos que la dirección será sólo 0x68 cuando el pin AD0 esté LOW y 0x69 cuando el pin AD0 esté HIGH. Esto aparece en el datasheet)
      Serial.print(F("Error leyendo sensor"));
      while (1);
    }

    delay(100); // Espera para que se estabilice el sensor

    /* Ángulos de inicio para el cálculo kalman y cálculo con giróscopos*/
    while (i2cRead(0x3B, i2cData, 6));         //Lectura de los Registros de Acelerómetros:0x3Bhex=59dec y 0x3Chex=60dec;  0x3Dhex=61dec y 0x3Ehex=62dec; 0x3Fhex=63dec y 0x40hex=64dec; y los mete en el vector i2cData ver pag.30 Mapa Registro pdf)
    accX = (i2cData[0] << 8) | i2cData[1];     //(i2cData[0] << 8)=esto es un desplazamiento de 8 bits a la izquierda de forma que el i2cData[0]queda en las posiciones más significativas y a la derecha quedan ceros.
    accY = (i2cData[2] << 8) | i2cData[3];     // Con la operación | se suman bit a bit las posiciones menos significatvas con el valor i2cData[1]. Así se obtiene un número
    accZ = (i2cData[4] << 8) | i2cData[5];     // de 16 bit a partir de dos números de 8 bit (porque los registros sólo pueden guardar 8 bits)

    // Source: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf eq. 25 and eq. 26
    // atan2 devuelve valores de -π a π (radianes)
    // Posteriormente se convierte de radianes a grados
    double roll  = atan2(accY, accZ) * RAD_TO_DEG;
    double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;

    // Inicializar ángulos para Kalman, filtro complementario y giroscopos.
    kalmanX.setAngle(roll);
    kalmanY.setAngle(pitch);
    gyroXangle = roll;
    gyroYangle = pitch;
    compAngleX = roll;
    compAngleY = pitch;
    timer = micros();

    /////// CÓDIGO PARA LA POSICION, VELOCIDAD Y ACELERACIÓN //////
    //Izquierdo
    pinMode(encoder0PinA, INPUT);
    digitalWrite(encoder0PinA, HIGH);       // Activar resistencias pullup
    pinMode(encoder0PinB, INPUT);
    digitalWrite(encoder0PinB, HIGH);       // Activar resistencias pullup
    //Recordar que son encoders de cuadratura, hay dos sensores por cada motor colocados a 90° uno del otro
    attachInterrupt(5, doEncoderA0, CHANGE);  // Encoder A en la interrupción 5 (pin 18)
    attachInterrupt(4, doEncoderB0, CHANGE);  // Encoder B en la interrupción 4 (pin 19)

    //Derecho
    pinMode(encoder1PinA, INPUT);
    digitalWrite(encoder1PinA, HIGH);       // Activar resistencias pullup
    pinMode(encoder1PinB, INPUT);
    digitalWrite(encoder1PinB, HIGH);       // Activar resistencias pullup
    //Recordar que son encoders de cuadratura, hay dos sensores por cada motor colocados a 90° uno del otro
    attachInterrupt(0, doEncoderA1, CHANGE);  // Encoder A en la interrupción 0 (pin 2)
    attachInterrupt(1, doEncoderB1, CHANGE);  // Encoder B en la interrupción 1 (pin 3)

    time1=millis();   // Inicializamos tiempo y posicion para el calculo de velocidad
    time2=millis();   // Inicializamos tiempo y posicion para el calculo de velocidad


//////////// MOTOR /////////////
//Para subir la frecuencia del PWM pin 7 del Mega 2560 de 489Hz a 31333Hz
TCCR4B = TCCR4B & 0b000 | 0x01;
//Para subir la frecuencia del PWM pin 9 del Mega 2560 de 489Hz a 31333Hz
TCCR2B = TCCR2B & 0b000 | 0x01;

pinMode(PinPWMD, OUTPUT);
pinMode(PinPWMI, OUTPUT);
pinMode(PinIn1I, OUTPUT);
pinMode(PinIn2I, OUTPUT);
pinMode(PinIn1D, OUTPUT);
pinMode(PinIn2D, OUTPUT);

///CONTROLADOR////
r=-1;
u=0;e=0;
////////////////////////////////

}
```

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

```
        dInput=kalAngleY-lastInput;
        u=Kp*e+ITerm-Kd*dInput;

        lastInput=kalAngleY;
        Ultimot=Ahora;
    }



    if (u<=0){
      digitalWrite(PinIn1I, HIGH);
      digitalWrite(PinIn2I, LOW);
      digitalWrite(PinIn1D, HIGH);
      digitalWrite(PinIn2D, LOW);
      PWMI=18+abs(u);
      PWMD=16+abs(u);}
    else {
      digitalWrite(PinIn1I, LOW);
      digitalWrite(PinIn2I, HIGH);
      digitalWrite(PinIn1D, LOW);
      digitalWrite(PinIn2D, HIGH);
      PWMI=19+abs(u);
      PWMD=17+abs(u);}

    if (PWMI>255){
      PWMI=255;
      PWMD=PWMI;}
    if (abs(e)>25){
      PWMI=0;
      PWMD=0;}
    analogWrite(PinPWMI, PWMI);
    analogWrite(PinPWMD, PWMD);

   ////////////////////////////////////////////////////////////////

    /* Enviar datos */
  #if 0 // Cambiar a 1 para activar
    Serial.print(accX); Serial.print("\t");
    Serial.print(accY); Serial.print("\t");
    Serial.print(accZ); Serial.print("\t");

    Serial.print(gyroX); Serial.print("\t");
    Serial.print(gyroY); Serial.print("\t");
    Serial.print(gyroZ); Serial.print("\t");

    Serial.print("\t");
  #endif

  #if 0 // Cambiar a 1 para activar
    Serial.print(roll); Serial.print("\t");
    Serial.print(gyroXangle); Serial.print("\t");
    Serial.print(compAngleX); Serial.print("\t");
    Serial.print(kalAngleX); Serial.print("\t");

    Serial.print("\t");

    Serial.print(pitch); Serial.print("\t");
    Serial.print(gyroYangle); Serial.print("\t");
    Serial.print(compAngleY); Serial.print("\t");
    Serial.print(kalAngleY); Serial.print("\t");
  #endif

  #if 0 // Cambiar a 1 para activar
    Serial.print("\t");
    double temperature = (double)tempRaw / 340.0 + 36.53; //Sensibilidad en Datasheet pag. 14
  (340LSB/°C) (offset=35°C segun del datasheet; pero es=36.53 en el pdf Mapa Registro)
    Serial.print(temperature); Serial.print("\t");
  #endif

  #if 0
    Serial.print("\t");

    Serial.print(encoder0Pos); Serial.print("\t");
    Serial.print(encoder1Pos); Serial.print("\t");
    Serial.print(velf); Serial.print("\t");
    Serial.print(velfd); Serial.print("\t");
    Serial.print(acel); Serial.print("\t");
    Serial.print(aceld); Serial.print("\t");

    Serial.print("\t");

    Serial.print(Kp); Serial.print("\t");
    Serial.print(Ki); Serial.print("\t");
    Serial.print(Kd); Serial.print("\t");

    Serial.print("\r\n");
    delay(2);

  #endif
  }

  ///// A CONTINUACIÓN VIENEN LAS FUNCIONES DE LAS INTERRUPCIONES PARA LOS ENCODERS ///////

  //INTERRUPCIONES ENCODERS MOTOR IZQUIERDO
  void doEncoderA0(){
    if (digitalRead(encoder0PinA) == HIGH) {   // busca un cambio de bajo-a-alto en el canal A
      if (digitalRead(encoder0PinB) == LOW) {   // comprueba el canal B para saber en qué sentido gira
        encoder0Pos++;}       // Horario
      else {encoder0Pos--;}   // Antihorario
    }
```

```
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
444         else {encoder1Pos++;}       // Antihorario
445       }
446
447       else {  // en otro caso sería un cambio de alto-a-bajo en canal B
448         if (digitalRead(encoder1PinA) == LOW) {   // comprueba el canal A para saber en qué sentido gira
449           encoder1Pos--;}           // Horario
450         else {encoder1Pos++;}       // Antihorario
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, 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 Ahora1, 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
108     /////// CÓDIGO PARA LAS INTERRUPCIONES //////
109     //Izquierdo
110     pinMode(encoder0PinA, INPUT);
111     digitalWrite(encoder0PinA, HIGH);
112     pinMode(encoder0PinB, INPUT);
113     digitalWrite(encoder0PinB, HIGH);
114     attachInterrupt(5, doEncoderA0, CHANGE);
115     attachInterrupt(4, doEncoderB0, CHANGE);
116     //Derecho
117     pinMode(encoder1PinA, INPUT);
118     digitalWrite(encoder1PinA, HIGH);
119     pinMode(encoder1PinB, INPUT);
120     digitalWrite(encoder1PinB, HIGH);
121     attachInterrupt(0, doEncoderA1, CHANGE);
122     attachInterrupt(1, doEncoderB1, CHANGE);
123     time1=millis();
124     time2=millis();
125     ////////////////////////////////////////////////
126
127     /////////// MOTOR /////////////
128     TCCR4B = TCCR4B & 0b000 | 0x01;
129     TCCR2B = TCCR2B & 0b000 | 0x01;
130     pinMode(PinPWMD, OUTPUT);
131     pinMode(PinPWMI, OUTPUT);
132     pinMode(PinIn1I, OUTPUT);
133     pinMode(PinIn2I, OUTPUT);
134     pinMode(PinIn1D, OUTPUT);
135     pinMode(PinIn2D, OUTPUT);
136
137     ///CONTROLADOR////
```

```
  phi_r=0;dphi_r=0;dtheta_r=0;
  u=0;
  Kp=0.2;
  Ki=0.1;
  ////////////////////////////////

  ///FILTRO VELOCIDAD////
  wf_2=1.0;
  wf_1=1.0;
  wf=1.0;
  vel_2=1.0;
  vel_1=1.0;
  wf0=1.0;
  ///////////////////////

  //CONFIGURACIÓN PARA MANIOBRAS//
  Tiempo0=millis();
  ////////////////////////////////
}


void loop() {

  /////CÓDIGO PARA LA IMU/////
  /* Actualización de todos los valores */
  while (i2cRead(0x3B, i2cData, 14));
  accX = ((i2cData[0] << 8) | i2cData[1]) - 432.0;
  accY = ((i2cData[2] << 8) | i2cData[3]) - 291.16;
  accZ = ((i2cData[4] << 8) | i2cData[5]) - 307.52;
  tempRaw = (i2cData[6] << 8) | i2cData[7];
  gyroX = (i2cData[8] << 8) | i2cData[9];
  gyroY = (i2cData[10] << 8) | i2cData[11];
  gyroZ = (i2cData[12] << 8) | i2cData[13];
  double dt = (double)(micros()- timer) / 1000000; // Cálculo de dt
  timer = micros();

  // atan2 devuelve valores de -π a π (radianes)
  // Posteriormente se convierte de radianes a grados
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
  double gyroXrate = gyroX / 131.0;
  double gyroYrate = gyroY / 131.0;

  if ((roll < -90 && kalAngleX > 90) || (roll > 90 && kalAngleX < -90)) {
    kalmanX.setAngle(roll);
    compAngleX = roll;
    kalAngleX = roll;
    gyroXangle = roll;
  } else
    kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);

  if (abs(kalAngleX) > 90)
    gyroYrate = -gyroYrate;
  kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);

  // Cálculo del ángulo mediante giróscopos sin ningún filtro
  gyroXangle += gyroXrate * dt;
  gyroYangle += gyroYrate * dt;
  // Cálculo del ángulo usando un filtro complementario
  compAngleX = 0.93 * (compAngleX + gyroXrate * dt) + 0.07 * roll;
  compAngleY = 0.93 * (compAngleY + gyroYrate * dt) + 0.07 * pitch;
 // Reseteo del ángulo de los giróscopos cuando se ha producido mucha deriva
  if (gyroXangle < -180 || gyroXangle > 180)
    gyroXangle = kalAngleX;
  if (gyroYangle < -180 || gyroYangle > 180)
    gyroYangle = kalAngleY;
  ////////////////////////////////




  ///////// CÓDIGO PARA POSICIÓN, VELOCIDAD Y ACELERACIÓN ////////
  if ((millis()-time1)>=10){
      veli=((encoder0Pos-pos0)*1000.0/(millis()-time1))*(PI/180.0);
      pos0=encoder0Pos;
      veld=((encoder1Pos-pos0d)*1000.0/(millis()-time1))*(PI/180.0);
      pos0d=encoder1Pos;
      vel=(veli+veld)/2;

      //Filtrado de la velocidad(BUTTERWORTH CON N=2; Wn=0.0784):
      wf=1.6544*wf_1 - 0.7059*wf_2 + 0.0129*vel + 0.0257*vel_1 + 0.0129*vel_2;
      wf_2=wf_1;
      wf_1=wf;
      vel_2=vel_1;
      vel_1=vel;

      acelf=(wf-wf0)*1000.0/(millis()-time1);
      wf0=wf;
      time1=millis();}
  ////////////////////////////////////////////////////////////////


  ///////CÓDIGO PARA MANIOBRAS DEL ROBOT ///////
  Tiempo1=millis();
  if ((Tiempo1-Tiempo0)<4000){
     dtheta_r=0;}
  else {if ((Tiempo1-Tiempo0)<9000){
          dtheta_r=3.8*PI;}
        else{ if ((Tiempo1-Tiempo0)<13000){
                  dtheta_r=0;}
```

```
              else{if ((Tiempo1-Tiempo0)<18000){
                      dtheta_r=-3.8*PI;}
                   else{dtheta_r=0;}
                  }
            }
      }
////////////////////////////////////////////


//////// CÓDIGO PARA CONTROLADOR ////////
Ahora=millis();
Cambiot=Ahora-Ultimot;

if (Cambiot>=40){
  phi=kalAngleY*(PI/180);
  dphi=gyroYrate*(PI/180);
  u=-332.8746*(phi_r-phi)-52.7835*(dphi_r-dphi)-3.1623*(dtheta_r-wf);
  r=-u*1.9;
  Ultimot=Ahora;}
Ahora1=millis();
Cambiot1=Ahora1-Ultimot1;

if (Cambiot1>=15){
  Ultimot1=Ahora1;
  e=r-acelf;
  ITerm += Ki*e;
  if((abs(ITerm))>255.0){
    if (ITerm>0) ITerm=255.0;
    else ITerm=-255.0;}
  u_m=Kp*e+ITerm;}


if (u_m<=0){
  digitalWrite(PinIn1I, HIGH);
  digitalWrite(PinIn2I, LOW);
  digitalWrite(PinIn1D, HIGH);
  digitalWrite(PinIn2D, LOW);
  PWMI=17+abs(u_m);
  PWMD=16+abs(u_m);}
else {
  digitalWrite(PinIn1I, LOW);
  digitalWrite(PinIn2I, HIGH);
  digitalWrite(PinIn1D, LOW);
  digitalWrite(PinIn2D, HIGH);
  PWMI=18+abs(u_m);
  PWMD=17+abs(u_m);}

if (PWMI>255){
  PWMI=255;
  PWMD=PWMI;}
if (abs(kalAngleY)>30){
  PWMI=0;
  PWMD=0;}
analogWrite(PinPWMI, PWMI);
analogWrite(PinPWMD, PWMD);
////////////////////////////////////////////////////////////////


/* ENVÍO DE DATOS */
#if 0 // Cambiar a 1 para activar
  Serial.print(r); Serial.print("\t");
  Serial.print(acel); Serial.print("\t");
  Serial.print(phi); Serial.print("\t");
  Serial.print(dphi); Serial.print("\t");
  Serial.print(vel); Serial.print("\t");
  Serial.print(millis()); Serial.print("\t");
  Serial.print("\t");
#endif
#if 0 // Cambiar a 1 para activar
  Serial.print(accX); Serial.print("\t");
  Serial.print(accY); Serial.print("\t");
  Serial.print(accZ); Serial.print("\t");

  Serial.print(gyroX); Serial.print("\t");
  Serial.print(gyroY); Serial.print("\t");
  Serial.print(gyroZ); Serial.print("\t");
  Serial.print("\t");
#endif
#if 0 // Cambiar a 1 para activar
  Serial.print(roll); Serial.print("\t");
  Serial.print(gyroXangle); Serial.print("\t");
  Serial.print(compAngleX); Serial.print("\t");
  Serial.print(kalAngleX); Serial.print("\t");
  Serial.print("\t");
  Serial.print(pitch); Serial.print("\t");
  Serial.print(gyroYangle); Serial.print("\t");
  Serial.print(compAngleY); Serial.print("\t");
  Serial.print(kalAngleY); Serial.print("\t");
#endif
#if 0 // Cambiar a 1 para activar
  Serial.print("\t");
  double temperature = (double)tempRaw / 340.0 + 36.53;
  Serial.print(temperature); Serial.print("\t");
#endif
#if 0
  Serial.print("\t");
  Serial.print(encoder0Pos); Serial.print("\t");
  Serial.print(encoder1Pos); Serial.print("\t");
  Serial.print(velf); Serial.print("\t");
  Serial.print(velfd); Serial.print("\t");
```

```
      Serial.print(acel); Serial.print("\t");
      Serial.print(aceld); Serial.print("\t");
      Serial.print("\t");
      Serial.print(Kp); Serial.print("\t");
      Serial.print(Ki); Serial.print("\t");
      Serial.print(Kd); Serial.print("\t");
    #endif
    #if 0 // Cambiar a 1 para activar
      Serial.print(u); Serial.print("\t");
      Serial.print(PWMI); Serial.print("\t");
      Serial.print(kalAngleY); Serial.print("\t");
    #endif
      Serial.print("\r\n");
      delay(1);
    //////////////////////////////////////
  }

  ///// A CONTINUACIÓN VIENEN LAS FUNCIONES DE LAS
  /// INTERRUPCIONES PARA LOS ENCODERS ///////
  //INTERRUPCIONES ENCODERS MOTOR IZQUIERDO
  void doEncoderA0(){
    if (digitalRead(encoder0PinA) == HIGH) {
      if (digitalRead(encoder0PinB) == LOW)
        encoder0Pos++;}       // Horario
      else {encoder0Pos--;}   // Antihorario
    }
    else{
      if (digitalRead(encoder0PinB) == HIGH) {
        encoder0Pos++;}          // Horario
      else {encoder0Pos--;}      // Antihorario
    }
   }
  void doEncoderB0(){
    if (digitalRead(encoder0PinB) == HIGH) {
      if (digitalRead(encoder0PinA) == HIGH) {
        encoder0Pos++;}          // Horario
      else {encoder0Pos--;}      // Antihorario
    }
    else {
      if (digitalRead(encoder0PinA) == LOW) {
        encoder0Pos++;}          // Horario
      else {encoder0Pos--;}      // Antihorario
    }
   }

   //INTERRUPCIONES ENCODERS MOTOR DERECHO
  void doEncoderA1(){
    if (digitalRead(encoder1PinA) == HIGH) {
      if (digitalRead(encoder1PinB) == LOW) {
        encoder1Pos--;}       // Horario
      else {encoder1Pos++;}   // Antihorario
    }
    else{
      if (digitalRead(encoder1PinB) == HIGH) {
        encoder1Pos--;}          // Horario
      else {encoder1Pos++;}      // Antihorario
    }
   }
  void doEncoderB1(){
    if (digitalRead(encoder1PinB) == HIGH) {
      if (digitalRead(encoder1PinA) == HIGH) {
  gira
        encoder1Pos--;}         // Horario
      else {encoder1Pos++;}     // Antihorario
    }

    else {
      if (digitalRead(encoder1PinA) == LOW) {
        encoder1Pos--;}          // Horario
      else {encoder1Pos++;}      // Antihorario
    }
   }
```

Archivo: CONTROLADOR_LQR_CON_BLUETOOTH.ino

```
  #include <Wire.h>
  #include "Kalman.h"
  Kalman kalmanX;
  Kalman kalmanY;

  /* PARA LOS DATOS DE LA IMU */
  double accX, accY, accZ;
  double gyroX, gyroY, gyroZ;
  int16_t tempRaw;
  double gyroXangle, gyroYangle;
  double compAngleX, compAngleY;
  double kalAngleX, kalAngleY;

  uint32_t timer;
  uint8_t i2cData[14];
```

```
  ////////////////////////////
  /* PARA LOS ENCODERS, VELOCIDAD, FILTRADO Y ACELERACIÓN */
  //Izquierdo
  #define encoder0PinA 18
  #define encoder0PinB 19
  volatile float encoder0Pos = 0;
  unsigned long time1;
  float pos0;
  float veli;

  //Derecho
  #define encoder1PinA 2
  #define encoder1PinB 3
  volatile float encoder1Pos = 0;
  unsigned long time2;
  float pos0d;
  float veld;

  //Filtro y acel
  float wf, wf_1, wf_2, wf0;
  float vel_1, vel_2, acelf;
  ////////////////////////////

  /* PARA EL MOTOR */
  #define PinPWMD 7
  #define PinPWMI 9
  #define PinIn1I 40
  #define PinIn2I 41
  #define PinIn1D 43
  #define PinIn2D 42
  float PWMD;
  float PWMI;
  ////////////////////////

  /* PARA EL CONTROL LQR*/
  float u;
  float phi_r,dphi_r,dtheta_r;
  float phi, dphi;
  float Ahora;
  float Cambiot, Ultimot;
  float Kp,Ki;
  float r,e;
  float Ahora1, Cambiot1, Ultimot1;
  float ITerm;
  float u_m;
  ////////////////////////

 /* PARA MANIOBRAS*/

 float u_mi, u_md;
 float tiempoDir;
 float Estado;

 /* PARA EL FILTRO DE LA VELOCIDAD*/

 float wf, wf_1, wf_2, wf0;
 float vel_1, vel_2, acelf;
 float veli, veld;


 /* PARA LAS VAR DE LA COMUNICACIÓN BLUETOOTH*/
 float K1, K2, K3, c; //variables para las constantes del LQR
 int Dir; //variable para la dirección (de 1 a 9)
 float xg, xt;
 int trim;

 float K1_r, K2_r, K3_r, c_r; //variables para las constantes en proceso de recepción del LQR
 int Dir_r; //variable para la dirección (de 1 a 9)
 float xg_r, xt_r;
 float trim_r;

 float Kp_r, Ki_r; //para la recepcion constantes control motor

 float tiempo0; //Para el cálculo de tiempo de bucle de programa.
 float bucle;

 /* PARA EL TRIMADO*/
 float trim_estado, adicion, trimado;
 float kalAngleY_trimado;
 ////////////////////////

void setup() {
  Serial.begin(115200);

  ////////// CÓDIGO IMU ////////////
  Wire.begin();
  TWBR = ((F_CPU / 400000L) - 16) / 2;
  i2cData[0] = 7
  i2cData[1] = 0x00;
  i2cData[2] = 0x00;
  i2cData[3] = 0x00;
  while (i2cWrite(0x19, i2cData, 4, false));
  while (i2cWrite(0x6B, 0x01, true));
  while (i2cRead(0x75, i2cData, 1));
  if (i2cData[0] != 0x68) {
    Serial.print(F("Error leyendo sensor"));
    while (1);
  }
  delay(100);
```

```
  /* Ángulos iniciales cálculo kalman y con giróscopos*/
  while (i2cRead(0x3B, i2cData, 6
  accX = (i2cData[0] << 8) | i2cData[1];
  accY = (i2cData[2] << 8) | i2cData[3];
  accZ = (i2cData[4] << 8) | i2cData[5];

  // Posteriormente se convierte de radianes a grados
  double roll =atan2(accY,accZ)*RAD_TO_DEG;
  double pitch=atan(-accX/sqrt(accY*accY+accZ*accZ))*RAD_TO_DEG;

  // Inicializar ángulos
  kalmanX.setAngle(roll);
  kalmanY.setAngle(pitch);
  gyroXangle = roll;
  gyroYangle = pitch;
  compAngleX = roll;
  compAngleY = pitch;
  timer = micros();
  ////////////////////////////////////////////////////


  /////// CÓDIGO PARA LAS INTERRUPCIONES //////
  //Izquierdo
  pinMode(encoder0PinA, INPUT);
  digitalWrite(encoder0PinA, HIGH);
  pinMode(encoder0PinB, INPUT);
  digitalWrite(encoder0PinB, HIGH);
  attachInterrupt(5, doEncoderA0, CHANGE);
  attachInterrupt(4, doEncoderB0, CHANGE);
  //Derecho
  pinMode(encoder1PinA, INPUT);
  digitalWrite(encoder1PinA, HIGH);
  pinMode(encoder1PinB, INPUT);
  digitalWrite(encoder1PinB, HIGH);
  attachInterrupt(0, doEncoderA1, CHANGE);
  attachInterrupt(1, doEncoderB1, CHANGE);
  time1=millis();
  time2=millis();
  ////////////////////////////////////////////

  /////////// MOTOR /////////////
  TCCR4B = TCCR4B & 0b000 | 0x01;
  TCCR2B = TCCR2B & 0b000 | 0x01;
  pinMode(PinPWMD, OUTPUT);
  pinMode(PinPWMI, OUTPUT);
  pinMode(PinIn1I, OUTPUT);
  pinMode(PinIn2I, OUTPUT);
  pinMode(PinIn1D, OUTPUT);
  pinMode(PinIn2D, OUTPUT);

    ///CONTROLADOR///
 phi_r=0;dphi_r=0;dtheta_r=0;
 u=0;
 Kp=0.2;//Kp=0.1;
 Ki=0.1;//Ki=0.4;
 K1=-332.8746; K2=-52.7835; K3=-3.1623; c=1.9; Dir=5.0; xg=1.0; xt=1.0;
 ////////////////////////////////


  ///FILTRO VELOCIDAD///
  wf_2=1.0;
  wf_1=1.0;
  wf=1.0;
  vel_2=1.0;
  vel_1=1.0;
  wf0=1.0;
  ///////////////////////

}


void loop() {

RecepcionBT();

  /////CÓDIGO PARA LA IMU/////
  /* Actualización de todos los valores */
  while (i2cRead(0x3B, i2cData, 14));
  accX = ((i2cData[0] << 8) | i2cData[1]) - 432.0;
  accY = ((i2cData[2] << 8) | i2cData[3]) - 291.16;
  accZ = ((i2cData[4] << 8) | i2cData[5]) - 307.52;
  tempRaw = (i2cData[6] << 8) | i2cData[7];
  gyroX = (i2cData[8] << 8) | i2cData[9];
  gyroY = (i2cData[10] << 8) | i2cData[11];
  gyroZ = (i2cData[12] << 8) | i2cData[13];
  double dt = (double)(micros()- timer) / 1000000; // Cálculo de dt
  timer = micros();

  // atan2 devuelve valores de -π a π (radianes)
  // Posteriormente se convierte de radianes a grados
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
  double gyroXrate = gyroX / 131.0;
  double gyroYrate = gyroY / 131.0;

  if ((roll < -90 && kalAngleX > 90) || (roll > 90 && kalAngleX < -90)) {
    kalmanX.setAngle(roll);
    compAngleX = roll;
    kalAngleX = roll;
```

```
      gyroXangle = roll;
    } else
      kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);

    if (abs(kalAngleX) > 90)
      gyroYrate = -gyroYrate;
    kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);

    // Cálculo del ángulo mediante giróscopos sin ningún filtro
    gyroXangle += gyroXrate * dt;
    gyroYangle += gyroYrate * dt;
    // Cálculo del ángulo usando un filtro complementario
    compAngleX = 0.93 * (compAngleX + gyroXrate * dt) + 0.07 * roll;
    compAngleY = 0.93 * (compAngleY + gyroYrate * dt) + 0.07 * pitch;
   // Reseteo del ángulo de los giróscopos cuando se ha producido mucha deriva
    if (gyroXangle < -180 || gyroXangle > 180)
      gyroXangle = kalAngleX;
    if (gyroYangle < -180 || gyroYangle > 180)
      gyroYangle = kalAngleY;
    ////////////////////////////////



    ///////// CÓDIGO PARA POSICIÓN, VELOCIDAD Y ACELERACIÓN ////////
    if ((millis()-time1)>=10){
        veli=((encoder0Pos-pos0)*1000.0/(millis()-time1))*(PI/180.0);
        pos0=encoder0Pos;
        veld=((encoder1Pos-pos0d)*1000.0/(millis()-time1))*(PI/180.0);
        pos0d=encoder1Pos;
        vel=(veli+veld)/2;

        //Filtrado de la velocidad(BUTTERWORTH CON N=2; Wn=0.0784):
        wf=1.6544*wf_1 - 0.7059*wf_2 + 0.0129*vel + 0.0257*vel_1 + 0.0129*vel_2;
        wf_2=wf_1;
        wf_1=wf;
        vel_2=vel_1;
        vel_1=vel;

        acelf=(wf-wf0)*1000.0/(millis()-time1);
        wf0=wf;
        time1=millis();}
    ////////////////////////////////////////////////////////////////


     ///////// CÓDIGO PARA CONTROLADOR ////////

      Ahora=millis();
      Cambiot=Ahora-Ultimot;
      Trimado();

      if (Cambiot>=40){
        phi=kalAngleY_trimado*(PI/180);
        dphi=gyroYrate*(PI/180);
        u=K1*(phi_r-phi) + K2*(dphi_r-dphi) + K3*(dtheta_r-wf);
          r=-u*c;    //Voy a tocar esto a ver si responde más rápido.
          Ultimot=Ahora;
      }

      Ahora1=millis();
      Cambiot1=Ahora1-Ultimot1;

      if (Cambiot1>=15){
        Ultimot1=Ahora1;
        e=r-acelf;
        ITerm += Ki*e;
          if((abs(ITerm))>255.0){
            if (ITerm>0) ITerm=255.0;
            else ITerm=-255.0;}
          u_m=Kp*e+ITerm;

      }

      Mando_Motor();

     ////////////////////////////////////////////////////////////////

      /*SECCIÓN PARA EL ENVÍO*/

      bucle=millis()-tiempo0;
      tiempo0=millis();

      EnvioBT();
     ////////////////////////////////////////////
  }

  ///// A CONTINUACIÓN VIENEN LAS FUNCIONES DE LAS
  /// INTERRUPCIONES PARA LOS ENCODERS ///////
  //INTERRUPCIONES ENCODERS MOTOR IZQUIERDO
  void doEncoderA0(){
    if (digitalRead(encoder0PinA) == HIGH) {
      if (digitalRead(encoder0PinB) == LOW)
        encoder0Pos++;}       // Horario
      else {encoder0Pos--;}   // Antihorario
    }
    else{
      if (digitalRead(encoder0PinB) == HIGH) {
        encoder0Pos++;}          // Horario
      else {encoder0Pos--;}      // Antihorario
    }
```

```
 }
void doEncoderB0(){
  if (digitalRead(encoder0PinB) == HIGH) {
    if (digitalRead(encoder0PinA) == HIGH) {
      encoder0Pos++;}          // Horario
    else {encoder0Pos--;}      // Antihorario
  }
  else {
    if (digitalRead(encoder0PinA) == LOW) {
      encoder0Pos++;}          // Horario
    else {encoder0Pos--;}      // Antihorario
  }
 }

 //INTERRUPCIONES ENCODERS MOTOR DERECHO
void doEncoderA1(){
  if (digitalRead(encoder1PinA) == HIGH) {
    if (digitalRead(encoder1PinB) == LOW) {
      encoder1Pos--;}        // Horario
    else {encoder1Pos++;}    // Antihorario
  }
  else{
    if (digitalRead(encoder1PinB) == HIGH) {
      encoder1Pos--;}          // Horario
    else {encoder1Pos++;}      // Antihorario
  }
 }
void doEncoderB1(){
  if (digitalRead(encoder1PinB) == HIGH) {
    if (digitalRead(encoder1PinA) == HIGH) {
gira
      encoder1Pos--;}          // Horario
    else {encoder1Pos++;}      // Antihorario
  }

  else {
    if (digitalRead(encoder1PinA) == LOW) {
      encoder1Pos--;}          // Horario
    else {encoder1Pos++;}      // Antihorario
  }
 }
//////////////////////////////////////////

///// A CONTINUACIÓN VIENE EL RESTO LAS FUNCIONES UTILIZADAS

void Mando_Motor(){
 switch(Dir){
      case 1:
      if ((millis()-tiempoDir)<1000){
        u_md=-u_m+10.0*xg;
        u_mi=-u_m-10.0*xg;
        Estado=10;}
      if((millis()-tiempoDir)>=1000){
        dtheta_r=0.76*PI*xt;
        u_mi=-u_m;
        u_md=u_mi;
        Estado=101;}
      break;

      case 2:
        dtheta_r=0.76*PI*xt;
        u_mi=-u_m;
        u_md=u_mi;
        Estado=20;
      break;

      case 3:
      if ((millis()-tiempoDir)<1000){
        u_mi=-u_m+10.0*xg;
        u_md=-u_m-10.0*xg;
        Estado=30;}
      if((millis()-tiempoDir)>=1000){
        dtheta_r=0.76*PI*xt;
        u_mi=-u_m;
        u_md=u_mi;
        Estado=301;}
      break;

      case 4:
        u_md=-u_m+10.0*xg;
        u_mi=-u_m-10.0*xg;
        Estado=40;
      break;

      case 5:
        dtheta_r=0;
        u_mi=-u_m;
        u_md=u_mi;
        Estado=50;
      break;

      case 6:
        u_mi=-u_m+10.0*xg;
        u_md=-u_m-10.0*xg;
        Estado=60;
      break;

      case 7:
      if ((millis()-tiempoDir)<1000){
        u_mi=-u_m+10.0*xg;
```

```
          u_md=-u_m-10.0*xg;
          Estado=70;}
        if((millis()-tiempoDir)>=1000){
          dtheta_r=-0.76*PI*xt;
          u_mi=-u_m;
          u_md=u_mi;
          Estado=701;}
        break;

        case 8:
          dtheta_r=-0.76*PI*xt;
          u_mi=-u_m;
          u_md=u_mi;
          Estado=80;
        break;

        case 9:
        if ((millis()-tiempoDir)<1000){
          u_md=-u_m+10.0*xg;
          u_mi=-u_m-10.0*xg;
          Estado=90;}
        if((millis()-tiempoDir)>=1000){
          dtheta_r=-0.76*PI*xt;
          u_mi=-u_m;
          u_md=u_mi;
          Estado=901;}
        break;

        default:
        dtheta_r=0;
          u_mi=-u_m;
          u_md=u_mi;
          Estado=112;
    }

     if (u_mi>=0){
       digitalWrite(PinIn1I, HIGH);
       digitalWrite(PinIn2I, LOW);
       PWMI=17+abs(u_mi);}
    else {
       digitalWrite(PinIn1I, LOW);
       digitalWrite(PinIn2I, HIGH);
       PWMI=18+abs(u_mi);}


     if (u_md>=0){
       digitalWrite(PinIn1D, HIGH);
       digitalWrite(PinIn2D, LOW);
       PWMD=16+abs(u_md);}
     else {
       digitalWrite(PinIn1D, LOW);
       digitalWrite(PinIn2D, HIGH);
       PWMD=17+abs(u_md);}

     if (PWMI>255){
       PWMI=255;}
     if (PWMD>255){
       PWMD=255;}

     if (abs(kalAngleY)>30){
       PWMI=0;
       PWMD=0;}

     analogWrite(PinPWMI, PWMI);
     analogWrite(PinPWMD, PWMD);

   }
   ////////////////////////////////////////



   void RecepcionBT(){
     while (Serial.available()>0)
       {
           xg_r = Serial.parseFloat(); //Esta es la velocidad de giro, la posición del slider.
           xt_r = Serial.parseFloat();
           trim_r = Serial.parseInt(); //Esta es la velocidad de traslacion, la posición del slider.
           Kp_r = Serial.parseFloat();
           Ki_r = Serial.parseFloat();
           K1_r = Serial.parseFloat();
           K2_r = Serial.parseFloat();
           K3_r = Serial.parseFloat();
           c_r = Serial.parseFloat();
           Dir_r = Serial.parseInt();

           //Cuando lea el carácter fin de línea ('\n') quiere decir que ha finalizado el envío
           if (Serial.read() == '\n')
           {
               //Actualizamos las variables reales si procede
                     if (K1_r != 0)
                       {Kp=Kp_r; Ki=Ki_r; K1=K1_r; K2=K2_r; K3=K3_r; c=c_r;}

                     if (Dir_r != 0)
                       {Dir=Dir_r;
                         if(Dir==1||Dir==3||Dir==7||Dir==9)
                           {tiempoDir=millis();}
                        }

                     if (xg_r != 0)
                       {xg=xg_r;}
```

```
                    if (xt_r != 0)
                      {xt=xt_r;}

                    if (trim_r != 0)
                      {trim=trim_r;
                       trim_estado=1;
                       }

            }
        }

}

void EnvioBT(){
  Serial.print("^");
  Serial.print(Kp); Serial.print(",");
  Serial.print(Ki); Serial.print(",");
  Serial.print(K1); Serial.print(",");
  Serial.print(K2); Serial.print(",");
  Serial.print(K3); Serial.print(",");
  Serial.print(c);Serial.print(",");
  Serial.print(bucle);Serial.print(","); //Tiempo que tarda en dar una vuelta el programa.
  Serial.print(Dir);Serial.print(",");
  Serial.print(Estado);Serial.print(",");
  Serial.print(xg);Serial.print(",");
  Serial.print(xt);Serial.print(",");
  Serial.print(trimado);
  delay(1);
}

//////////////////////////////////////////

void Trimado(){
  if (trim_estado==1){
          switch(trim){
            case 1:
            adicion =-0.5;   //Cada vez que pulsamos uno de los botones de trimar en la aplicación android subimos o bajamos medio grado.
            break;

            case 2:
            adicion = 0.5;
            break;

            default:
            adicion=0;
          }
     trim_estado=0;
  }
  else {adicion=0;}
   trimado+=adicion;
   kalAngleY_trimado= kalAngleY+trimado;
}
```