FACULTAD DE INGENIERIA
Universidad de Buenos Aires

(TA135) Taller de Automatización y Control
1er Cuatrimestre de 2024

**Actividad 2**

**09 de abril de 2024**


| | | |
|---|---|---|
| Facundo Arballo | – Padrón 105096 – | farballo@fi.uba.ar |
| Francisco Spaltro | – Padrón 102098 – | fspaltro@fi.uba.ar |



**Resumen**

Programar el controlador del tanque cónico en Arduino, utilizando, si es posible, los métodos forward y backwards difference, y bilineal. El controlador que obtuvimos en tiempo continuo es

$$C(s) = -3{,}1623\frac{s + 0{,}00237}{s}$$


# 1. Forward difference

Desarrollando el método de discretización por *forward difference*, se tiene que

$$s = \frac{z-1}{T}$$

Por lo tanto

$$\begin{aligned} C_d(z) &= -k\frac{\frac{z-1}{T} + 0{,}00237}{\frac{z-1}{T}} \\ &= -k\frac{z + (0{,}00237\ T - 1)}{z-1} \\ &= -k\frac{1 + k_1\ z^{-1}}{1 - z^{-1}} \end{aligned}$$

donde $k_1 = 0{,}00237\ T - 1$
Entonces,

$$u(z)(1 - z^{-1}) = -ke(z)(1 + k_1\ z^{-1})$$

Escribiendo la ecuación en diferencias, resulta:

$$\begin{aligned} u(n) - u(n-1) &= -ke(n) - k\ k_1\ e(n-1) \\ u(n) &= -ke(n) - k\ (0{,}00237\ T - 1)\ e(n-1) + u(n-1) \end{aligned}$$

## 2. Backward difference

Desarrollando el método de discretización por *backward difference*, se tiene que

$$s = \frac{z-1}{zT}$$

Por lo tanto

$$\begin{aligned} C_d(z) &= -k\frac{\frac{z-1}{zT} + 0{,}00237}{\frac{z-1}{zT}} \\ &= -k\frac{z-1+0{,}00237\ T\ z}{z-1} \\ &= -k\frac{k_1 z - 1}{z-1} \\ &= -k\frac{k_1 - z^{-1}}{1-z^{-1}} \end{aligned}$$

Siendo $k_1 = 1 + 0{,}00237\ T$. Luego:

$$(1-z^{-1})u(z) = -k(k_1 - z^{-1})e(z)$$

Expresándolo en su ecuación en diferencias:

$$\begin{aligned} u(n) - u(n-1) &= -k\ k_1\ e(n) + k\ e(n-1) \\ u(n) &= -k\ (1 + 0{,}00237\ T)\ e(n) + k\ e(n-1) + u(n-1) \end{aligned}$$

## 3. Bilineal

Desarrollando el método de discretización bilineal, se tiene que

$$s = \frac{2}{T}\frac{z-1}{z+1}$$

Por lo tanto

$$
\begin{aligned}
C_d(z) &= -k\frac{\frac{2}{T}\frac{z-1}{z+1} + 0{,}00237}{\frac{2}{T}\frac{z-1}{z+1}} \\
&= -k\frac{\frac{z-1}{z+1} + \frac{T}{2}0{,}00237}{\frac{z-1}{z+1}} \\
&= -k\frac{(z-1) + \frac{T}{2}0{,}00237\ (z+1)}{z-1} \\
&= -k\frac{z(1+\frac{T}{2}0{,}00237) + \frac{T}{2}0{,}00237\ -1}{z-1} \\
&= -k\frac{(1+\frac{T}{2}0{,}00237) + (\frac{T}{2}0{,}00237\ -1)z^{-1}}{1-z^{-1}}
\end{aligned}
$$

Luego:

$$
(1-z^{-1})u(z) = -k\left[\left(1+\frac{T}{2}0{,}00237\right) + \left(\frac{T}{2}0{,}00237\ -1\right)z^{-1}\right]e(z)
$$

Expresándolo en su ecuación en diferencias:

$$
\begin{aligned}
u(n) - u(n-1) &= -k\left(1+\frac{T}{2}0{,}00237\right)e(n) + k\left(1-\frac{T}{2}0{,}00237\right)e(n-1) \\
u(n) &= -k\left(1+\frac{T}{2}0{,}00237\right)e(n) + k\left(1-\frac{T}{2}0{,}00237\right)e(n-1) + u(n-1)
\end{aligned}
$$

# 4. Código

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

uint16_t tiempo_inicial = 0;
uint16_t tiempo_final = 0;

float offset_giro_x = -0.04;
float offset_giro_y = 0.03;
float offset_giro_z = 0;

float offset_accel_x = 1.13;
float offset_accel_y = 0.39;
float offset_accel_z = -0.75;

float angulo_gir_x = 0;
float angulo_accel_x = 0;
float angulo_x = 0;
```

```
21
22 float alpha = 0.98;
23
24 const long int periodo_lectura = 10000;
25
26 void setup(void) {
27   Serial.begin(115200);
28
29   if (!mpu.begin()) {
30     Serial.println("Failed to find MPU6050 chip");
31     while (1) {
32       delay(10);
33     }
34   }
35   Serial.println("MPU6050 Found!");
36
37   mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
38
39     mpu.setGyroRange(MPU6050_RANGE_500_DEG);
40   mpu.setFilterBandwidth(MPU6050_BAND_10_HZ);
41
42   delay(100);
43 }
44
45 void loop() {
46   tiempo_inicial = micros();
47
48   obtener_angulo_giroscopo_x();
49   obtener_angulo_accel_x();
50
51   angulo_x = alpha*angulo_gir_x + (1-alpha)*angulo_accel_x;
52
53   Serial.print(" Angulo giroscopio X: ");
54   Serial.print(angulo_gir_x);
55   Serial.print(" Angulo acelerometro X: ");
56   Serial.print(angulo_accel_x);
57   Serial.print(" Angulo X: ");
58   Serial.println(angulo_x);
59   tiempo_final = micros();
60   delayMicroseconds(periodo_lectura - (tiempo_final - tiempo_inicial));
61 }
62
63 void obtener_angulo_giroscopo_x() {
64   sensors_event_t a, g, temp;
65   mpu.getEvent(&a, &g, &temp);
66
67   angulo_gir_x = angulo_x + (g.gyro.x-offset_giro_x) * (periodo_lectura/1e6) *
       180 / M_PI;
68 }
69
70 void obtener_angulo_accel_x() {
71   sensors_event_t a, g, temp;
72   mpu.getEvent(&a, &g, &temp);
73
74   angulo_accel_x  = atan2(a.acceleration.y-offset_accel_y , a.acceleration.z-
       offset_accel_z) * 180 / M_PI;
```

```
75 }
```