

9 février 2018

Présentation du Robot Lego

Qualid SRHRI – Julien BONNARDEL – Bertrand E. TALAKI

Le projet

Implémenter une commande multi-objective
suiveur de ligne et anti-collision



Loi de commande d'orientation

- Dérivée de la vitesse d'orientation

$$\theta' = \frac{V_g - V_d}{L}$$

- Fonction de transfert de θ

$$H_{\theta}(s) = \frac{\theta(s)}{u_{\theta}} = \frac{1}{L*s}$$

- Correction proportionnel intégral

$$C_{\theta}(s) = \frac{k_{i\theta}}{s} + k_{p\theta}$$

Loi de commande d'orientation

- Fonction de transfert du système en boucle fermée

$$H(s) = \frac{C_{\theta}(s)*H_{\theta}(s)}{1+C_{\theta}(s)*H_{\theta}(s)} = \frac{\frac{k_p\theta}{l}s + \frac{k_i\theta}{l}}{s^2 + \frac{k_p\theta}{l}s + \frac{k_i\theta}{l}}$$

Discrétisation

- Calcul de la pulsation critique
 - Fonction margin
- Période d'échantillonnage

$$\omega_c T \in [0.05, 0.14]$$

- Méthode de Tustin

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

Stabilité

- Changement de variable bilinéaire

$$z = \frac{1+w}{1-w}$$

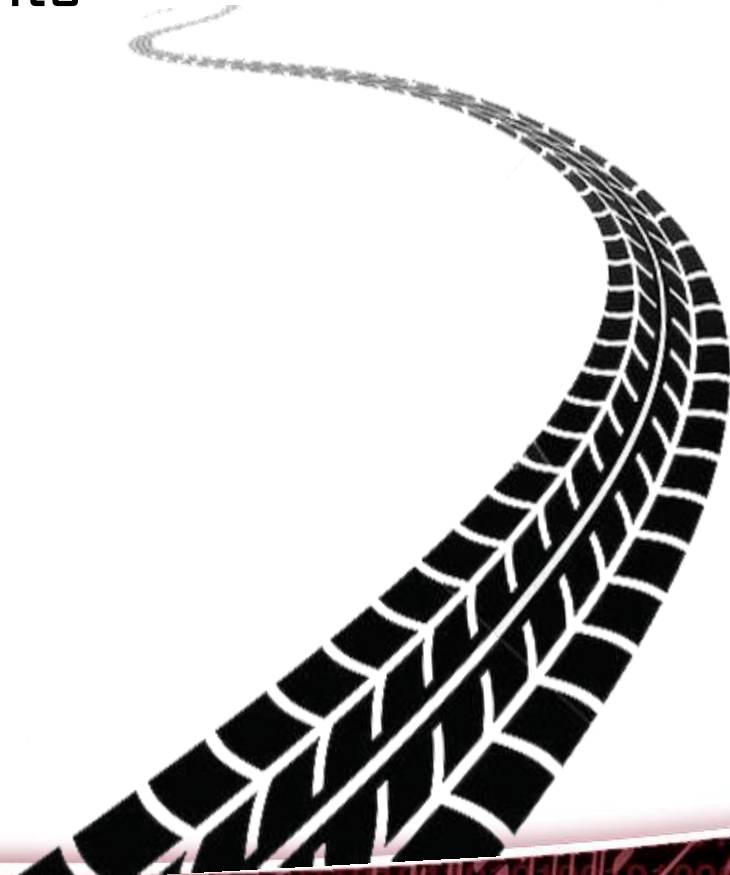
- Polynôme caractéristique du système

$$\pi(w) = \frac{4}{T^2} * w^2 + \frac{2k_p\theta}{TL} * w + \frac{k_i\theta}{L}$$

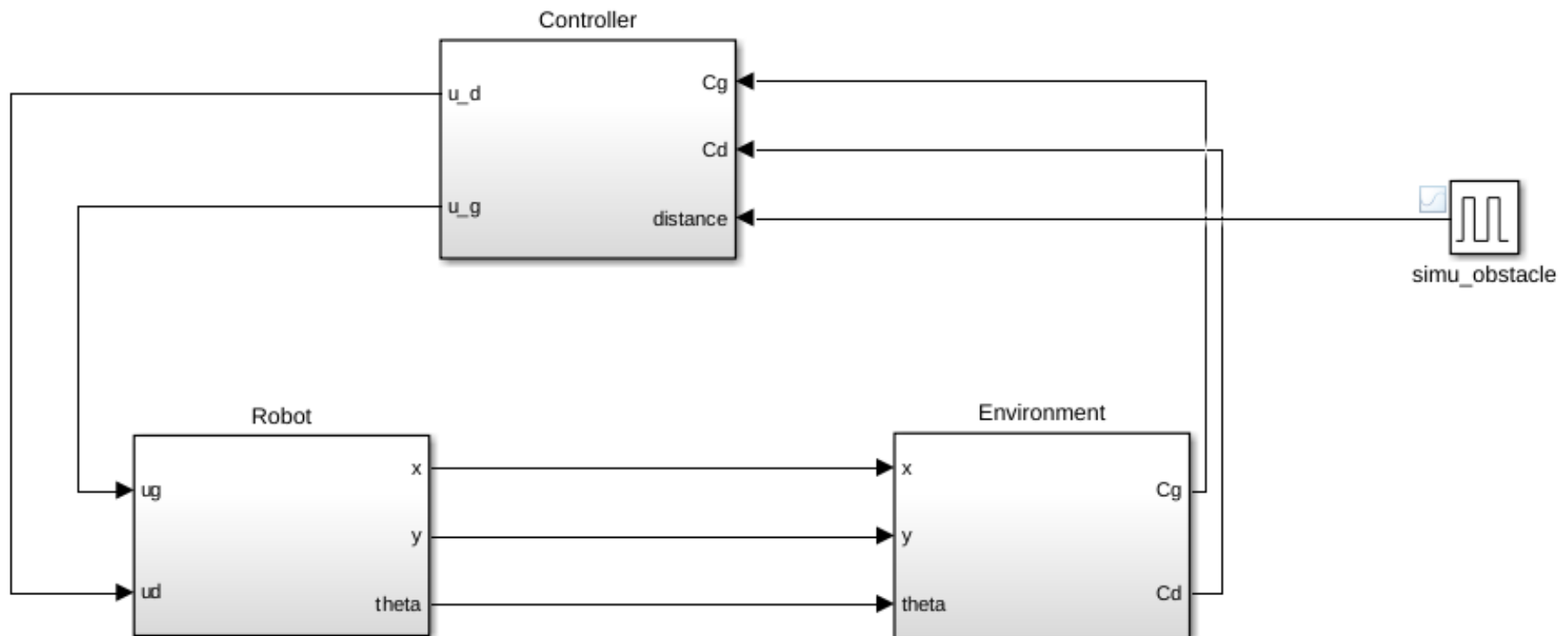
L'implémentation

Simulink

- Suiveur de ligne implémenté
 - Capteur de lumière
- Evitement d'obstacle
 - Réduire la vitesse
 - Faire demi-tour



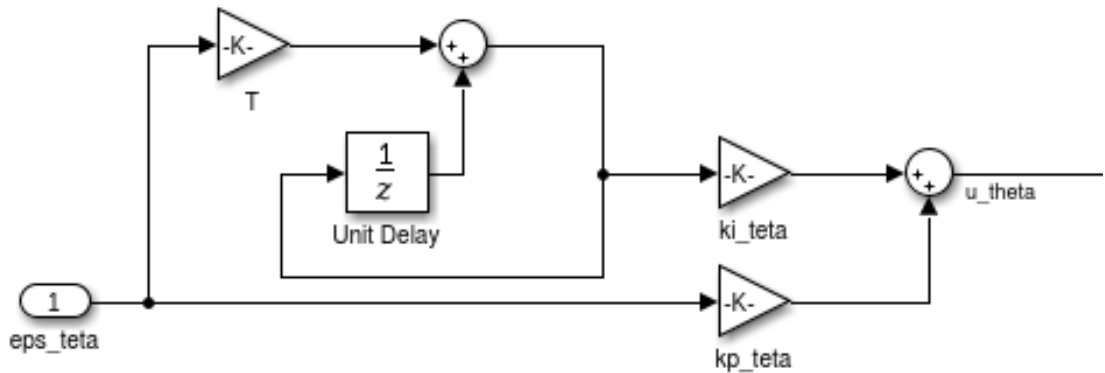
L'implémentation Simulink



L'implémentation Simulink

➤ Bloc Correcteurs

- Ajout de la fonction de transfert de Tustin



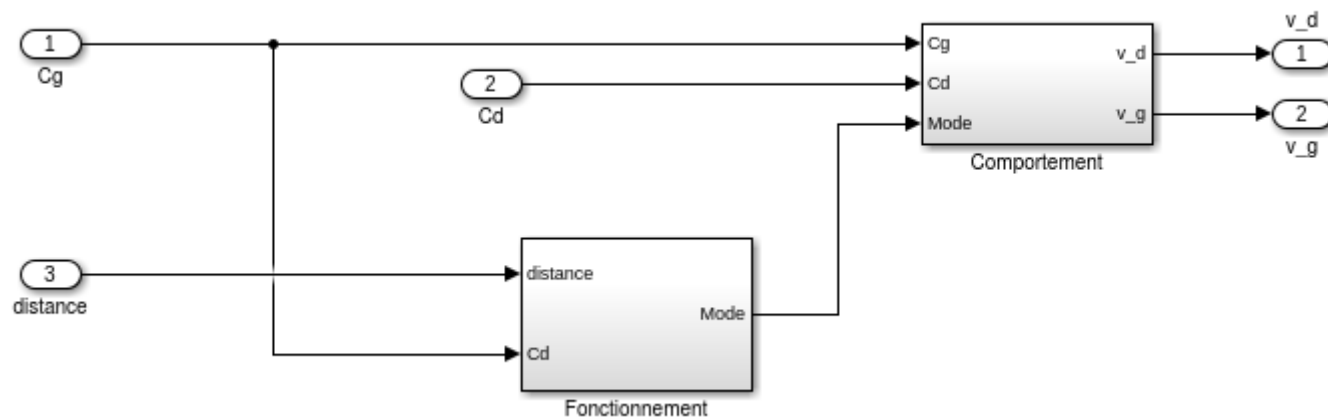
L'implémentation Simulink

➤ Fonctionnement

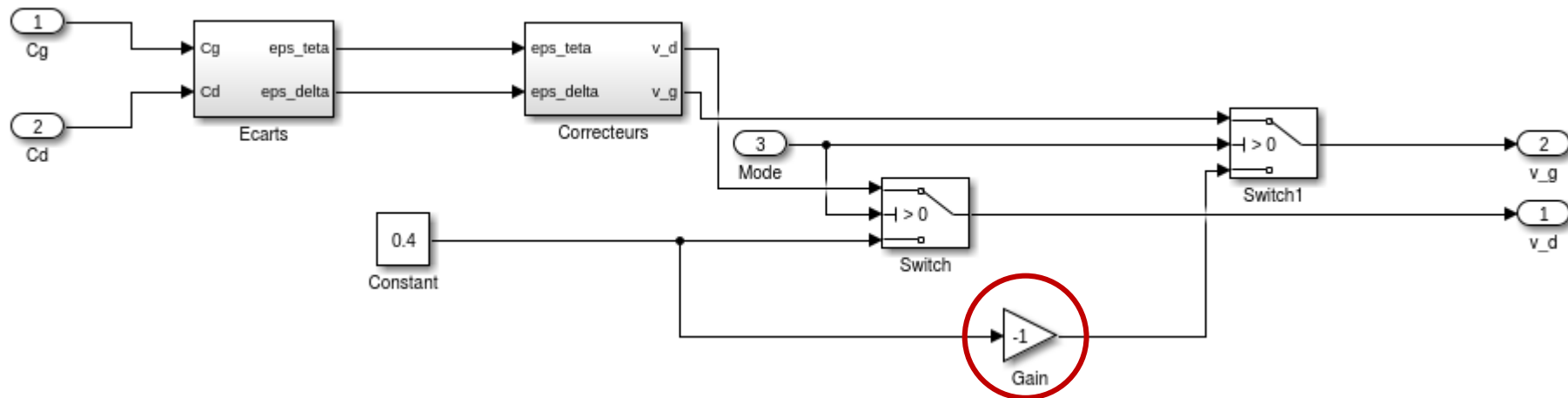
- Mode suiveur de ligne
- Mode évitement d'obstacle

➤ Comportement

- Suiveur : aucun changement
- Evitement : réduction de la vitesse + demi tour



L'implémentation Simulink

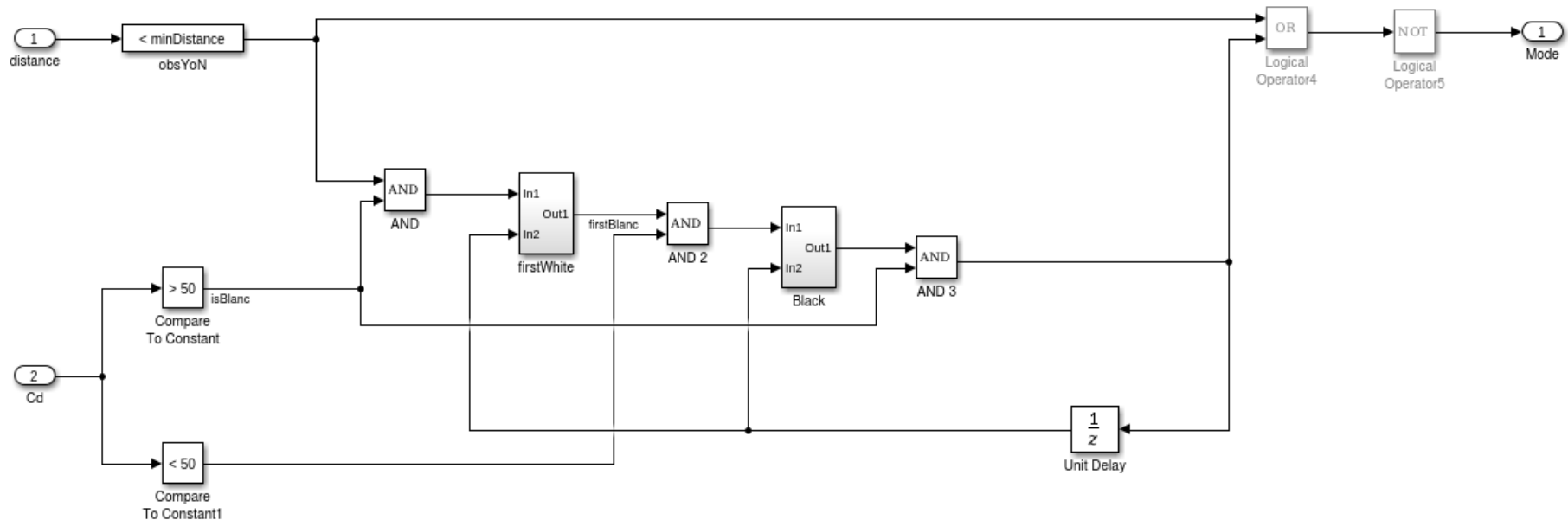


Permet le demi - tour

➤ Comportement

- Vitesse du suiveur de ligne
- Vitesse diminué + rotation

L'implémentation Simulink



➤ Fonctionnement

- Mode suiveur de ligne
- Mode évitement d'obstacle

L'implémentation

Code

```
258 /*-----
259  Speed and mode Functions
260 -----
261 - Control engine right and left
262 - Change mode when there is an obstacle
263 -----*/
264
265 void speedMotor(U8 port_id, _real speed, int place){
266     int speed_cast_to_int = (int) (speed * 100);
267
268     if( speed_cast_to_int > speedMax ){
269         speed_cast_to_int = speedMax;
270     }
271
272     if ( speed_cast_to_int < -speedMax ){
273         speed_cast_to_int = -speedMax;
274     }
275
276     display_goto_xy(0, place);
277     display_int(speed_cast_to_int,3);
278     display_update();
279
280     return ecrobot_set_motor_speed(port_id, speed_cast_to_int);
281 }
282
```

```
230 /*-----
231  Others Function
232 -----
233 - Need to grade in [0;100]
234 - Check distance
235 -----*/
236
237 _real grade( _real black, _real white, long value ){
238     _real gradeValue = (black - value) / (black - white) * 100;
239
240     if ( gradeValue > 100 ){
241         return 100;
242     } else if ( gradeValue < 0 ) {
243         return 0;
244     } else {
245         return gradeValue;
246     }
247
248 }
```


L'implémentation

Code

```
66 /* Init and terminate OSEK */
67 void ecrobot_device_initialize() {
68     /*
69      * HERE: put here specific code that will be executed ONCE
70      * when the application starts
71      * TYPICALLY: initialization of (light) sensors
72      */
73     ecrobot_init_sonar_sensor(NXT_PORT_S3); //Distance
74     ecrobot_set_light_sensor_active(NXT_PORT_S2); //Gauche
75     ecrobot_set_light_sensor_active(NXT_PORT_S1); //Droite
76 }
77
78 void ecrobot_device_terminate() {
79     /*
80      * HERE: put here specific code that will be executed ONCE
81      * when the application stops
82      * TYPICALLY: finalization of (light) sensors
83      */
84     ecrobot_term_sonar_sensor(NXT_PORT_S3); //Distance
85     ecrobot_set_light_sensor_inactive(NXT_PORT_S2); //Gauche
86     ecrobot_set_light_sensor_inactive(NXT_PORT_S1); //Droite
87 }
```

```
102 void usr_init(){
103     // Init du buffer
104     bufferMode[0] = true;
105     bufferMode[1] = true;
106
107     GetResource(lcd);
108
109     display_clear(0);
110     display_goto_xy(0,0);
111     display_string("White init");
112
113     // Init les capteurs sur le blanc
114     while(!ecrobot_is_ENTER_button_pressed()) {
115         leftW = ecrobot_get_light_sensor(NXT_PORT_S2) + 20;
116         rightW = ecrobot_get_light_sensor(NXT_PORT_S1) + 20;
117
118         display_clear(1);
119         display_goto_xy(0, 0);
```

L'implémentation

Code

```
320 TASK(HighTask) { // Comportement
321     if( count == 0 ){
322         currentMode = (currentMode + 1) % 2;
323     }
324     display_goto_xy(0,0);
325
326     if( bufferMode[currentMode % 2] ){
327         display_clear(1);
328         display_goto_xy(1,6);
329         display_string("Go ahead");
330     } else {
331         display_clear(1);
332         display_goto_xy(1,3);
333         display_string("Obstacle");
334         display_goto_xy(1,7);
335         display_string("Turn !");
336     }
337 }
```

```
283 void Comportement_0_v_d(_real speed){
284     speedMotor(NXT_PORT_A, speed, 1);
285 }
286
287 void Comportement_0_v_g(_real speed){
288     speedMotor(NXT_PORT_B, speed, 2);
289 }
290
291
292
293 void Fonctionnement_0_Mode(_boolean mode){
294     bufferMode[(currentMode + 1) % 2] = mode;
295 }
```

```
306 TASK(LowTask) { // Fonctionnement
307
308     display_goto_xy(0,0);
309     display_string("Block Mode");
310     display_update();
311
312     Fonctionnement_I_Cd_n(grade(rightB, rightW, ecrebot_get_light_sensor
(NXT_PORT_S1)));
313     Fonctionnement_I_distance(checkDistance(ecrobot_get_sonar_sensor(NXT_PORT_S3)));
314
315     Fonctionnement_step();
316
317     TerminateTask();
318 }
```

Merci pour votre attention !



DEMO