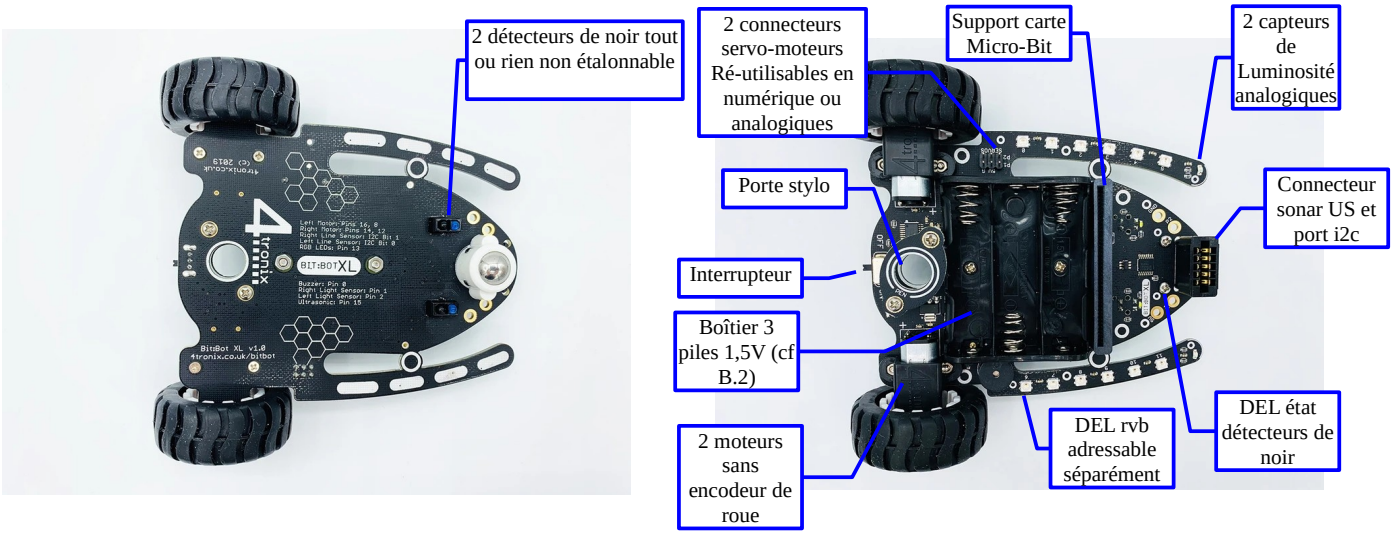
Tests Bit:Bot XL

Table des matières

A. Constitution du Robot



B. Pré-requis

I. Installation sous Ubuntu 20.04

L'installation se fait avec la commande « sudo apt-get install mu-editor », le site en ligne MakeCode permet aussi de programmer ce robot avec des bibliothèques adaptées.

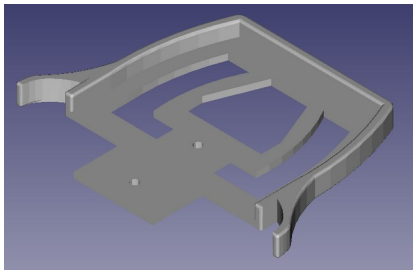
II. Alimentation du robot

Initialement, le robot est vendu avec un compartiment de 3 piles AA (3 x 1,5 V). Si vous utilisez des

batteries 1,2 V à leur place, le capteur Sonar ne fonctionnera pas.

Nous avons donc remplacé le compartiment 3 piles AA par un compartiment 4 piles AA avec l'accord du fournisseur.

III. Pare-choc



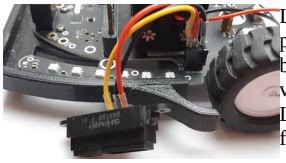
Les branches avant du robot sont fragiles, afin d'éviter qu'un élève puisse les casser en saisissant le robot. D'où l'utilisation d'un pare-choc réalisé avec une imprimante 3D.

C. Ports

N° port	Fonction	Programmes test	Remarques
16, 8	Moteur gauche	**Vitesse fixe (max ou arrêté)** Gauche avant vitesse max : *pin16.write_digital(0)* *pin8.write_digital(1)* Gauche arrière vitesse max : *pin16.write_digital(1)* *pin8.write_digital(0)* **Vitesse variable** Gauche arrière : *pin16.write_digital(0)* *pin8.write_analog(1023)* → Commence à tourner à 70 et maximum à 1023 Gauche avant : *pin16.write_digital(1)* *pin8.write_analog(930)* → Commence à tourner à 930 et maximum à 0	Pin vitesse : 8 Pin direction : 16
14, 12	Moteur droit	Droite avant vitesse max : *pin14.write_digital(1)* *pin12.write_digital(0)* Droite arrière vitesse max : *pin14.write_digital(0)* *pin12.write_digital(1)* **Vitesse variable** Droite arrière : *pin14.write_digital(0)* *pin12.write_analog(200)* → Commence à tourner à 70 et maximum à 1023 Droite avant : *pin14.write_digital(1)* *pin12.write_analog(930)* *→ Commence à tourner à 930 et maximum à 0*	Pin vitesse : 12 Pin direction : 14

i2c bit 0 et 1	Suiveur de ligne	*from microbit import ** *import neopixel* *I2CADDR = 0x1c # address of PCA9557* *fireleds = neopixel.NeoPixel(pin13, 12)* *def getLine(bit):* *mask = 1 << bit* *value = 0* *try:* *value = i2c.read(I2CADDR, 1)[0]* *except OSError:* *pass* *if (value & mask) > 0:* *return 1* *else:* *return 0* *while True:* *if(getLine(0) == 0): # noir détecté à gauche, led blanche éteinte* *fireleds[5]=(40,0,0)* *else:* *fireleds[5]=(0,0,40) # non noir détecté à gauche, led blanche allumée* *if(getLine(1) == 0):* *fireleds[11]=(40,0,0)# noir détecté à droite, led blanche éteinte* *else:* *fireleds[11]=(0,40,0) non noir détecté à droite, led blanche allumée* *fireleds.show()* *sleep(200)*	La valeur retournée est binaire « 0 » noir détecté, sinon « 1 ».
13	Leds RVB	*from microbit import ** *import neopixel* *while True:* *np = neopixel.NeoPixel(pin13, 12)* *np[2] = (40, 0, 40)* *np.show( )*	Numéro de la led choisie : np[n°] → n° compris entre 0 et 11 Composante rvb : np[2]=(r,v,b) → valeurs comprises entre 0 et 255 0, 0, 0 : éteint ; 255,255,255 : blanc
0	Buzzer	*from microbit import ** *while True:* *pin0.write_digital(1)* *display.show("S")* *sleep(400)* *pin0.write_digital(0)* *display.clear()* *sleep(400)*	
1, 2	Capteur lumière droit et gauche	*from microbit import ** *import neopixel* *fireleds = neopixel.NeoPixel(pin13, 12)* *while True:* *leftVal = pin2.read_analog() # Lecture capteur gauche* *fireleds[5]=(40,0,0)* *fireleds.show()*	La valeur retournée est analogique codée sur 10 bits.

		`` display.show("@") sleep(1000) display.show(leftVal) sleep(2000) fireleds[5]=(0,0,0) fireleds.show() rightVal = pin1.read_analog() # Lecture capteur droite fireleds[11]=(40,0,0) fireleds.show() display.show("@") sleep(1000) display.show(rightVal) sleep(2000) fireleds[11]=(0,0,0) fireleds.show() ```	
15	Capteur Ultrasons	``` from microbit import * from utime import ticks_us, sleep_us SONAR = pin15 def sonar( ): SONAR.write_digital(1) # Envoi une impulsion de 10us sleep_us(10) SONAR.write_digital(0) SONAR.set_pull(SONAR.NO_PULL) while SONAR.read_digital() == 0: # Vérifie que l'impulsion a bien été envoyée start = ticks_us() # temps de départ de l'impulsion while SONAR.read_digital() == 1: # Vérifie que l'impulsion a bien été retournée (écho) end = ticks_us() # temps de retour de l'impulsion echo = end-start distance = int(0.01715 * echo) # calcul de la distance en cm return distance while True: display.show("@") display.scroll(sonar()) sleep(1000) ```	
P1, P1	Commandes PWM/MLI pour servomoteurs	``` from microbit import * def setServoAngle(pin, angle): if (angle >= 0 and angle <= 180): pin.write_analog(26 + (angle*102)/180) else: raise ValueError("Servomotor angle have to be set between 0 and 180") while True: setServoAngle(pin1, 0) sleep(1000) setServoAngle(pin1, 90) sleep(1000) ```	

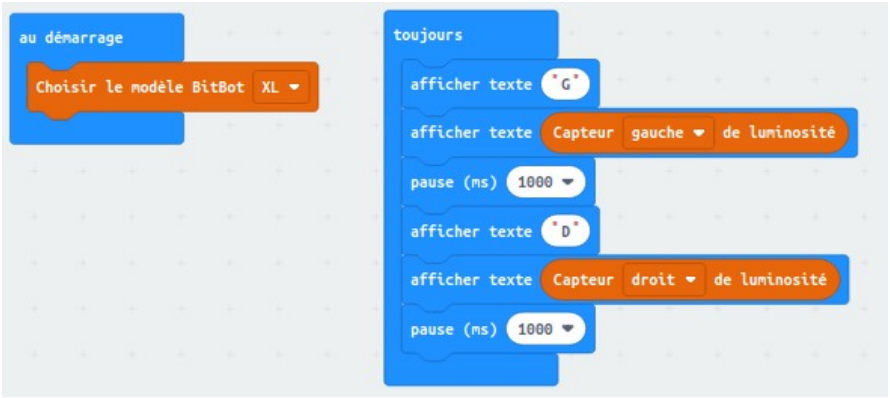
		setServoAngle(pin1, 180) *sleep(1000)*	
P1, P1	Acquisitions analogiques	 Les capteurs de distance Sharp sont connectés sur les ports P1 et P2 du robot initialement destinés aux branchement des servomoteurs. Ils renvoient une valeur analogique. La valeur renvoyée est forte si l'obstacle est proche et faible s'il est éloigné. *from microbit import ** *while True:* *distance=pin1.read_analog()* *display.scroll(distance)* *sleep(1000)*	

D. Exemples de programmes

I. Capteurs lumière

```
bitbot.select_model(BBModel.XL)

def on_forever():
basic.show_string("G")
basic.show_string("" + str((bitbot.read_light(BBLightSensor.LEFT))))
basic.pause(1000)
basic.show_string("D")
basic.show_string("" + str((bitbot.read_light(BBLightSensor.RIGHT))))
basic.pause(1000)
basic.forever(on_forever)
```



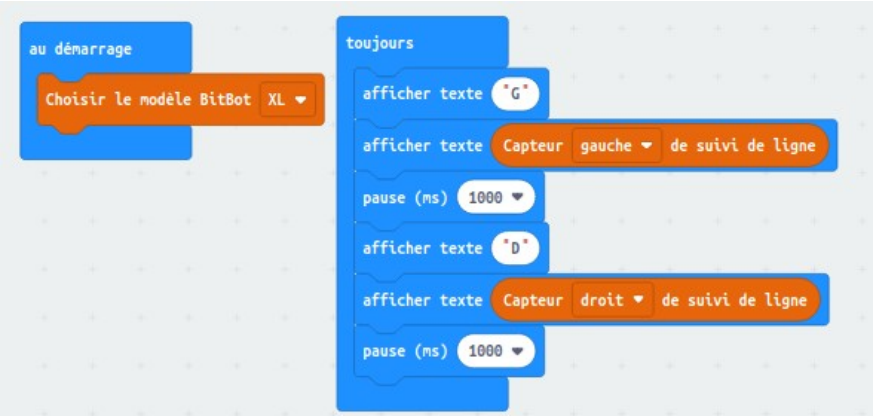
Résultats sur 10 bits

II. Suiveur de ligne

```
bitbot.select_model(BBModel.XL)

def on_forever():
basic.show_string("G")
basic.show_string("" + str((bitbot.read_line(BBLineSensor.LEFT))))
```

```
basic.pause(1000)
basic.show_string("D")
basic.show_string("" + str((bitbot.read_line(BBLineSensor.RIGHT))))
basic.pause(1000)
basic.forever(on_forever)
```



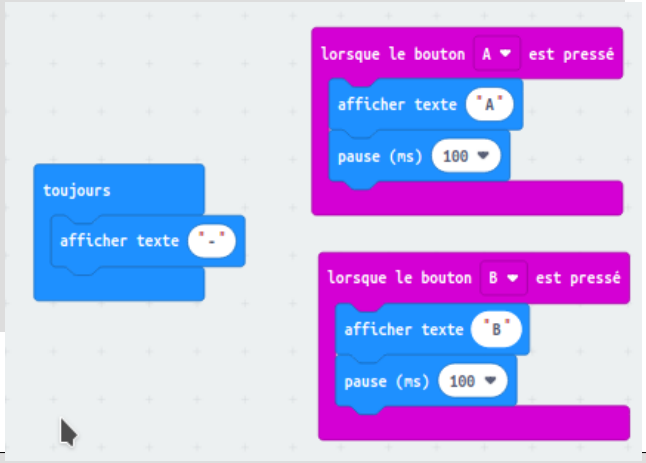
Retour binaire, si noir retour « 1 » si blanc retour « 0 »

III. Boutons

```
def on_button_pressed_a():
basic.show_string("A")
basic.pause(100)
input.on_button_pressed(Button.A, on_button_pressed_a)

def on_button_pressed_b():
basic.show_string("B")
basic.pause(100)
input.on_button_pressed(Button.B,
on_button_pressed_b)

def on_forever():
basic.show_string("-")
basic.forever(on_forever)
```



E. Extensions « maison »

I. Capteurs de proximité latéraux analogiques

Voir plus haut

II. Extension « i2c »

De la gauche vers la droite vue de l'avant du connecteur du capteur ultrason : masse, pin 15, SCL, SDA, 3V3