

```

1 from spike import PrimeHub, LightMatrix, Button, StatusLight, ForceSensor, MotionSensor, Speaker, ColorSensor, App,
  DistanceSensor, Motor, MotorPair
2 from spike.control import wait_for_seconds, wait_until, Timer
3 from spike.operator import equal_to, greater_than_or_equal_to, less_than_or_equal_to
4 from math import trunc, fabs, copysign
5 from utime import ticks_ms
6 import gc
7
8
9 hub = PrimeHub()
10
11 hub.light_matrix.show_image('HAPPY')
12
13 driveMotors = MotorPair('B', 'A')
14
15 attachmentMotors = MotorPair('C', 'D')
16
17 leftAttachment = Motor('C')
18 rightAttachment = Motor('D')
19
20 leftDrive = Motor('B')
21 rightDrive = Motor('A')
22
23 leftSensor = ColorSensor('E')
24 rightSensor = ColorSensor('F')
25
26 gc.collect()
27
28 #Just driving. Set the amount in degrees, steering (0 is straight), and power
29 def drive(amount, steering, power):
30     #setup
31     hub.light_matrix.show_image('TORTOISE')
32     driveMotors.set_stop_action('brake')
33     #execution
34     driveMotors.move(amount, 'degrees', steering, power)
35     driveMotors.stop()
36
37 #Wait for a button push, with a fancy clock
38 def wait():
39     #setup
40     driveMotors.set_stop_action('brake')
41     attachmentMotors.set_stop_action('brake')
42     clockFaces = ['CLOCK12',
43 'CLOCK1','CLOCK2','CLOCK3','CLOCK4','CLOCK5','CLOCK6','CLOCK7','CLOCK8','CLOCK9','CLOCK10','CLOCK11']
44     done = False
45     index = 0
46     clockSpeed = 30
47     hub.left_button.was_pressed()
48     #execution
49     driveMotors.stop()
50     attachmentMotors.stop()
51     while not done:
52         hub.light_matrix.show_image(clockFaces[trunc(index/clockSpeed)% len(clockFaces)])
53         index += 1
54         if hub.left_button.was_pressed():
55             done = True

```

```
56 def alignWithLine():
57     hub.light_matrix.show_image('DUCK')
58     #Find the line
59     #FindTheLine setup
60     done = False
61     driveMotors.set_stop_action('brake')
62     driveSpeed = 15
63     backUp = -8
64     rightSensorValue = 100
65     leftSensorValue = 100
66     darkEnough = 30
67     isRightBlack = False
68     #FindTheLine action
69     while not done:
70         driveMotors.start(0, driveSpeed)
71         rightSensorValue = rightSensor.get_reflected_light()
72         leftSensorValue = leftSensor.get_reflected_light()
73         if rightSensorValue < darkEnough or leftSensorValue < darkEnough:
74             done = True
75             if rightSensorValue < darkEnough:
76                 isRightBlack = True
77                 hub.light_matrix.show_image('ARROW_NW')
78             else:
79                 isRightBlack = False
80                 hub.light_matrix.show_image('ARROW_SW')
81     driveMotors.stop()
82     wait_for_seconds(1)
83     #driveMotors.move(backUp, 'degrees', driveSpeed)
84     #The second sensor needs to find black
85     #SecondSensor setup
86     done = False
87     smallerDriveSpeed = ((-driveSpeed) * 1)
88     #SecondSensor action
89     if isRightBlack:
90         while not done:
91             driveMotors.start_tank(driveSpeed, trunc(smallerDriveSpeed))
92             leftSensorValue = leftSensor.get_reflected_light()
93             if leftSensorValue < darkEnough:
94                 done = True
95     else:
96         while not done:
97             driveMotors.start_tank(trunc(smallerDriveSpeed), driveSpeed)
98             rightSensorValue = rightSensor.get_reflected_light()
99             if rightSensorValue < darkEnough:
100                 done = True
101     driveMotors.stop()
102     #WIGGLE time
103     hub.light_matrix.show_image('PACMAN')
104     #Wiggle set-up
105     time = 300
106     numberOfLoops = 0
107     targetValue = 61
108     #Wiggle action
109     while numberOfLoops < time:
110         numberOfLoops += 1
111         Ck = 0.5
112         rightSensorValue = rightSensor.get_reflected_light()
```

```
113     leftSensorValue = leftSensor.get_reflected_light()
114     rightDriveAmount = (rightSensorValue - targetValue) * Ck
115     leftDriveAmount = (leftSensorValue - targetValue) * Ck
116     driveMotors.start_tank(trunc(leftDriveAmount), trunc(rightDriveAmount))
117     driveMotors.stop()
118
119
120 #Turns right a set amount of degrees, not TO a specific degree
121 #Note: Turning right goes from 0 to 179, then -179 to 0 in degrees
122 def turnRightForward(endAngle, power):
123     #setup
124     hub.motion_sensor.reset_yaw_angle()
125     hub.light_matrix.show_image('ARROW_N')
126     driveMotors.set_stop_action('brake')
127     #execution
128     rightDrive.stop()
129     #Negative powers counter clockwise
130     leftDrive.start(-power)
131     wait_until(hub.motion_sensor.get_yaw_angle, greater_than_or_equal_to, endAngle)
132     leftDrive.stop()
133
134 #Turns left a set amount of degrees, not TO a specific degree
135 #Note: Turning left goes from 179 to 0, then 0 to -179 in degrees
136 def turnLeftForward(endAngle, power):
137     #setup
138     hub.motion_sensor.reset_yaw_angle()
139     hub.light_matrix.show_image('ARROW_N')
140     driveMotors.set_stop_action('brake')
141     #execution
142     leftDrive.stop()
143     #Negative powers counter clockwise
144     rightDrive.start(power)
145     wait_until(hub.motion_sensor.get_yaw_angle, less_than_or_equal_to, -endAngle)
146     rightDrive.stop()
147
148 #Turns right a set amount of degrees, not TO a specific degree, but it's backwards
149 def turnRightBack(endAngle, power):
150     #setup
151     hub.motion_sensor.reset_yaw_angle()
152     #wait_for_seconds(0.7)
153     hub.light_matrix.show_image('ARROW_N')
154     driveMotors.set_stop_action('brake')
155     #execution
156     leftDrive.stop()
157     #Negative powers counter clockwise
158     rightDrive.start(-power)
159     wait_until(hub.motion_sensor.get_yaw_angle, greater_than_or_equal_to, endAngle)
160     rightDrive.stop()
161
162 #Turns left a set amount of degrees, not TO a specific degree, but it's backwards
163 def turnLeftBack(endAngle, power):
164     #setup
165     hub.motion_sensor.reset_yaw_angle()
166     #wait_for_seconds(0.7)
167     hub.light_matrix.show_image('ARROW_N')
168     driveMotors.set_stop_action('brake')
169     #execution
```

```
170 rightDrive.stop()
171 #Negative powers counter clockwise
172 leftDrive.start(power)
173 wait_until(hub.motion_sensor.get_yaw_angle, less_than_or_equal_to, -endAngle)
174 leftDrive.stop()
175
176
177 #Turns right a set amount of degrees using tank, not TO a specific degree
178 #Note: Turning right goes from 0 to 179, then -179 to 0 in degrees
179 def turnRightTank(endAngle, power):
180     #setup
181     hub.motion_sensor.reset_yaw_angle()
182     hub.light_matrix.show_image('DUCK')
183     driveMotors.set_stop_action('brake')
184     steering = 100
185     #execution
186     currentAngle = hub.motion_sensor.get_yaw_angle()
187     while currentAngle <= endAngle:
188         currentAngle = hub.motion_sensor.get_yaw_angle()
189         driveMotors.start_tank(power, -power)
190     driveMotors.stop()
191
192 #Turns left a set amount of degrees using tank, not TO a specific degree
193 #Note: Turning left goes from 179 to 0, then 0 to -179 in degrees
194 def turnLeftTank(endAngle, power):
195     #setup
196     hub.motion_sensor.reset_yaw_angle()
197     hub.light_matrix.show_image('DUCK')
198     driveMotors.set_stop_action('brake')
199     steering = -100
200     count = 0
201     #execution
202     currentAngle = hub.motion_sensor.get_yaw_angle()
203     while currentAngle >= -endAngle:
204         currentAngle = hub.motion_sensor.get_yaw_angle()
205         driveMotors.start_tank(-power, power)
206     driveMotors.stop()
207
208 #Finds black on the board
209 def findBlackTurning(steering, power, lightIntensity, lightSensor):
210     #setup
211     hub.light_matrix.show_image('SKULL')
212     driveMotors.set_stop_action('brake')
213     #steering=0
214     #execution
215     driveMotors.start(steering, power)
216     wait_until(lightSensor.get_reflected_light, less_than_or_equal_to, lightIntensity)
217     driveMotors.stop()
218
219 def findBlack(power, lightIntensity, lightSensor):
220     #setup
221     hub.light_matrix.show_image('SKULL')
222     driveMotors.set_stop_action('brake')
223     steering=0
224     #execution
225     driveMotors.start(steering, power)
226     wait_until(lightSensor.get_reflected_light, less_than_or_equal_to, lightIntensity)
```

```

227 driveMotors.stop()
228
229
230
231 def driveStraightForSeconds(power, timeInSeconds):
232     #-177 is limit
233     #setup
234     goal = 0
235     hub.light_matrix.show_image('GHOST')
236     hub.motion_sensor.reset_yaw_angle()
237     done = False
238     Kp = 3
239     timer = Timer()
240     timer.reset()
241     yaw = hub.motion_sensor.get_yaw_angle()
242     #execution
243     while not done:
244         yaw = hub.motion_sensor.get_yaw_angle()
245         error = goal-yaw
246         P=Kp*error
247         driveMotors.start(P, power)
248         if timeInSeconds < timer.now():
249             driveMotors.stop()
250             done=True
251
252 def timeFollower1(lightSensor, driveTime, speed, direction):
253     #Good numbers below
254     #Kp for proportional only is 2.2
255     #Kp = 1.1, ki = 0.02, kd = 20, iClamp = 1000, ifClamp = 15, oClamp = 65
256     #8/27 good numbers: speed=25 steering=0, goal=68, kp=1.3, ki=0.1, kd=10, iClamp=99, ifClamp=30, oClamp=65, dClamp=80
257
258     #PREPARATIONS!!
259     #Speed, steering and goal
260     #speed = 20 #15 #20 #25 #20 #15 #10 #35 #40 #45 #40 #35 #30 #25 #30 #25 #30 #40 #25
261     steering = 0
262     goal = 68
263     percent = goal
264     startTime = 0
265     endTime = 0
266
267     #Errors
268     error = 0
269     totalError = 0
270     lastError = 0
271
272     #Defining
273     pFix = 0
274     iFix = 0
275     dFix = 0
276     overall = 0
277
278     #The Ks
279     #Kp only is 2
280     #kp = 1.9
281     #kp = 0.8, kd = 0.9
282     Kp = 1.3 #0.9 #1 #0.9 #1.4 #1 #0.9 #0.8 #0.7 #1 #0.9 #1.8 #1.9 #1.75 #1.5 #1.4 #2 #2.1 #2.2 #2 #1.3 #1.1 #0.9 #0.8 #1.2 #1.1
283     #1.3
284     Ki = 0.1 #0.025 #0 #0.025 #0 #0.025 #0.05 #0.1 #0.05 #0.1 #0.1 #0.2 #0.02 #0.002

```

```

284   Kd = 10 #0.5 #80 #100 #1 #0.5 #0.25 #0.5 #1 #2 #4 #5 #10 #5 #1.1 #1 #0.9 #0.8 #0.75 #0.5 #1 #2 #5 #15 #10 #10 #15 #20 #10
      #25 #20 #10 #15
285
286   #Clamping
287   iClamp = 99 #1000 #99 #1000
288   ifClamp = 30 #15 #30 #15 #30 #15
289   oClamp = 65
290   dClamp = 80 #40 #80 #40 #99
291
292   #Time Stuff
293   sampleTime = 23 #14
294   #Moved to ms, are using ticks_ms
295   #Old time values are in seconds 0.01 #0.004 #0.005 #0.004 #0.003 #0.004 #0.005 #0.002 #0.003 #0.006 #0.001 #0.005 #0.01
      #0.001
296   timer = Timer()
297   driveEndTime = 0
298
299   if lightSensor == leftSensor:
300       if direction == 'inside':
301           directionMult = -1
302       elif direction == 'outside':
303           directionMult = 1
304   elif lightSensor == rightSensor:
305       if direction == 'inside':
306           directionMult = 1
307       if direction == 'outside':
308           directionMult = -1
309
310   #EXECUTE!!
311   #Image
312   hub.light_matrix.show_image('GIRAFFE')
313
314   #Start the motors!
315   driveMotors.start(steering, speed)
316
317   #Print so we know when it starts
318   #print('*****')
319
320   timer.reset()
321
322   #Loop!
323
324   done = False
325   #driveTime is being multiplied by 1000 because driveTime is seconds and ticks_ms is milliseconds
326   driveEndTime = ticks_ms() + driveTime * 1000
327   while not done:
328       startTime = ticks_ms()
329       #Recording light reflectiveness
330       percent = lightSensor.get_reflected_light()
331
332       #Proportional
333       # there is no need for if statements because if percent is more then the robot will go
334       # left and if percent is less then itll go right
335       error = goal - percent
336       pFix = Kp * (error)
337
338       #Integral
339       totalError = totalError + error

```

```

340     if fabs(totalError) > iClamp:
341         totalError = copysign(iClamp, totalError)
342     iFix = totalError * Ki
343     if fabs(iFix) > ifClamp:
344         iFix = copysign(ifClamp, totalError)
345
346     #Derivative
347     dFix = (error - lastError) * Kd
348     if fabs(dFix) > dClamp:
349         dFix = copysign(dClamp, dFix)
350     lastError = error
351
352     #Overall
353     overall = pFix + iFix + dFix
354     if fabs(overall) > oClamp:
355         overall = copysign(oClamp, overall)
356
357     #Start
358     driveMotors.start(trunc(overall * directionMult), speed)
359     endTime = ticks_ms()
360     #print(startTime - endTime, ', ')
361     #wait_for_seconds(time)
362     #Printing for troubleshooting
363     #print('totalError:', totalError, ' iFix:', iFix, ' dFix:', dFix, ' pFix:', pFix, ' overall:', overall)
364
365
366     if ticks_ms() > driveEndTime:
367         hub.light_matrix.show_image('DIAMOND')
368         done = True
369
370     while ticks_ms() < startTime + sampleTime:
371         pass
372
373
374
375 def followLineForTime(lightSensor, driveTime, direction, chooseStop):
376     timeFollower1(lightSensor, 0.5, 20, direction) #1 2
377     timeFollower1(lightSensor, driveTime - 0.5, 30, direction) #32 #30 #35
378     driveMotors.set_stop_action(chooseStop)
379     driveMotors.stop()
380
381 def lineFollower1(lightSensor, driveTime, speed, direction):
382     #Good numbers below
383     #Kp for proportional only is 2.2
384     #Kp = 1.1, ki = 0.02, kd = 20, iClamp = 1000, ifClamp = 15, oClamp = 65
385     #8/27 good numbers: speed=25 steering=0, goal=68, kp=1.3, ki=0.1, kd=10, iClamp=99, ifClamp=30, oClamp=65, dClamp=80
386
387     #PREPARATIONS!!
388     #Speed, steering and goal
389     #speed = 20 #15 #20 #25 #20 #15 #10 #35 #40 #45 #40 #35 #30 #25 #30 #25 #30 #40 #25
390     steering = 0
391     goal = 68
392     percent = goal
393
394     #Errors
395     error = 0
396     totalError = 0

```

```

397 lastError = 0
398
399 #Defining
400 pFix = 0
401 iFix = 0
402 dFix = 0
403 overall = 0
404
405 #The Ks
406 #Kp only is 2
407 #kp = 1.9
408 #kp = 0.8, kd = 0.9
409 Kp = 1.3 #0.9 #1 #0.9 #1.4 #1 #0.9 #0.8 #0.7 #1 #0.9 #1.8 #1.9 #1.75 #1.5 #1.4 #2 #2.1 #2.2 #2 #1.3 #1.1 #0.9 #0.8 #1.2 #1.1
#1.3
410 Ki = 0.1 #0.025 #0 #0.025 #0 #0.025 #0.05 #0.1 #0.05 #0.1 #0.1 #0.2 #0.02 #0.002
411 Kd = 10 #0.5 #80 #100 #1 #0.5 #0.25 #0.5 #1 #2 #4 #5 #10 #5 #1.1 #1 #0.9 #0.8 #0.75 #0.5 #1 #2 #5 #15 #10 #10 #15 #20 #10
#25 #20 #10 #15
412
413 #Clamping
414 iClamp = 99 #1000 #99 #1000
415 ifClamp = 30 #15 #30 #15 #30 #15
416 oClamp = 65
417 dClamp = 80 #40 #80 #40 #99
418
419 #Time Stuff
420 sampleTime = 23 #20 #36 #25 #50 #40 #30 #14
421 #Moved to ms, are using ticks_ms
422 #Old time values 0.004 #0.005 #0.004 #0.003 #0.004 #0.005 #0.002 #0.003 #0.006 #0.001 #0.005 #0.01 #0.001
423 timer = Timer()
424
425 #Inside and Outside
426 if lightSensor == leftSensor:
427     if direction == 'inside':
428         directionMult = -1
429     elif direction == 'outside':
430         directionMult = 1
431 elif lightSensor == rightSensor:
432     if direction == 'inside':
433         directionMult = 1
434     if direction == 'outside':
435         directionMult = -1
436
437 #Light sensor stuff
438 if lightSensor == leftSensor:
439     light_stop_sensor = rightSensor
440 elif lightSensor == rightSensor:
441     light_stop_sensor = leftSensor
442
443 #EXECUTE!!
444 #Image
445 hub.light_matrix.show_image('YES')
446
447 #Start the motors!
448 driveMotors.start(steering, speed)
449
450 #Timer Reset
451 timer.reset()
452

```



```
453 #Loop!
454 done = False
455 #driveTime is being multiplied by 1000 because driveTime is seconds and ticks_ms is milliseconds
456 driveEndTime = ticks_ms() + driveTime * 1000
457 while not done:
458     startTime = ticks_ms()
459
460     #Recording light reflectiveness
461     percent = lightSensor.get_reflected_light()
462     percentStop = light_stop_sensor.get_reflected_light()
463
464     #Proportional
465     # there is no need for if statements because if percent is more then the robot will go
466     # left and if percent is less then itll go right
467     error = goal - percent
468     pFix = Kp * (error)
469
470     #Integral
471     totalError = totalError + error
472     if fabs(totalError) > iClamp:
473         totalError = copysign(iClamp, totalError)
474     iFix = totalError * Ki
475     if fabs(iFix) > ifClamp:
476         iFix = copysign(ifClamp, totalError)
477
478     #Derivative
479     dFix = (error - lastError) * Kd
480     if fabs(dFix) > dClamp:
481         dFix = copysign(dClamp, dFix)
482     lastError = error
483
484     #Overall
485     overall = pFix + iFix + dFix
486     if fabs(overall) > oClamp:
487         overall = copysign(oClamp, overall)
488
489     #Start
490     driveMotors.start(trunc(overall * directionMult), speed)
491     endTime = ticks_ms()
492     #wait_for_seconds(time)
493
494     #Stopping for other sensor seeing black
495     if percentStop <= 30:
496         hub.light_matrix.show_image('ARROW_W')
497         done = True
498
499     #Waiting for time then ending the loop
500     if ticks_ms() > driveEndTime:
501         hub.light_matrix.show_image('DIAMOND')
502         done = True
503
504     while ticks_ms() < startTime + sampleTime:
505         pass
506
507
508 #Defining Line Follower
509 def followLineUntilNextLine(lightSensor, driveTime, direction, chooseStop):
```

```

510 lineFollower1(lightSensor, 0.5, 20, direction) #1 2
511 lineFollower1(lightSensor, driveTime - 0.5, 30, direction) #32 #35
512 driveMotors.set_stop_action(chooseStop)
513 driveMotors.stop()
514
515
516 def lineFollowTesterer():
517     drive(120, 0, 50) #Forward drive before we find the line
518     #driveMotors.move_tank(147, 'degrees', 30, 0) #90 degree turn right for line
519     turnRightForward(86, 30) #Turn towards the line before we find it
520     findBlack(20, 25, rightSensor) #Drive froward to find the line and stops
521     followLineForTime(rightSensor, 9.375, 'outside', 'brake') #Follows the line until after the line in the wheel flip mission
522     #driveMotors.stop()
523     #wait()
524     followLineUntilNextLine(rightSensor, 3, 'outside', 'brake') #Follows the line until detects the line near the treadmill
525
526 def moveRake(upOrDown, rakeTime, rakePower):
527     if upOrDown == 'up':
528         rakePower = -rakePower
529     elif upOrDown == 'down':
530         rakePower = rakePower
531     else:
532         print('Incorrect phrasing, make sure it is either up or down')
533     rightAttachment.start(rakePower)
534     wait_for_seconds(rakeTime)
535     rightAttachment.stop()
536
537 def moveRakeForDegrees(rakeDegrees, rakePower):
538     rightAttachment.run_for_degrees(-rakeDegrees, rakePower)
539
540 def moveLeftForDegrees(leftDegrees, leftPower):
541     leftAttachment.run_for_degrees(-leftDegrees, leftPower)
542
543 def moveLeft(leftSeconds, leftPower):
544     leftAttachment.run_for_seconds(leftSeconds, -leftPower)
545
546 #
547 # | (A function goes in a program) ^
548 # | Programs below|Functions above ^
549 # | Programs below|Functions above/ \
550 # | Programs below|Functions above/ \
551 # | Programs below|Functions above | |
552 # | | Programs below|Functions above | |
553 # \ /Programs below|Functions above | |
554 # \ /Programs below|Functions above | |
555 # \ Programs below|Functions above | |
556 # \ (A program is made up of functions)| |
557 #
558
559
560 #robot run 2
561 def rakeForklift():
562     drive(400, 0, 50)
563     moveRake('up', 1, 20)
564     drive(-200, 0, 50)
565
566 #rakeForklift()

```

```

567 #moveRake('down', 0.6)
568
569
570
571
572 def boxDropOff():
573     drive(-200, 0, 30)
574     moveRakeForDegrees(50, 50)
575     drive(200, 0, 30)
576     moveRakeForDegrees(100, 50)
577     drive(50, 0, 30)
578     moveRakeForDegrees(210, 50)
579
580 #
581 #
582 #
583 #
584 #
585 #
586 #
587 #
588 #
589 #
590 #
591 #
592 #
593 #
594
595 def oneDelivery():
596     #driving forward from launch area
597     drive(400,0,45) #360 370 380 400 375
598     #turns right
599     turnRightForward(48,30) #55 #45
600     wait_for_seconds(0.2)
601     #drives forward to cargo connect area
602     #wait()
603     drive(770,0,45) #700 580 800 750 650 760
604     #turns toward the cargo connect area
605     turnRightForward(20,30) #24 #26 #18 20 #16
606     #drives into the cargo connect area
607     #wait()
608     findBlack(45, 40, leftSensor)
609     drive(475, 0, 45) #750 800 700 800 850 900 1200 775 575
610     #lines up to drop off inovative attachment
611     turnRightForward(53,30) #53 60 63 43 85
612     #drives into cargo connect area
613     drive(40, 0, 40) #50 100 130 100
614     #releases the innovative attachment into the cargo connect area
615     moveRakeForDegrees(440,55)#140
616     #backs out of cargo connect area
617     drive(162, 0, -40) #142 100 165 185 200
618     #turnRightTank(15,15)
619     #turnLeftTank(15,15)
620     #wait_for_seconds(0.2)#wait()
621     #turns left to go foward to the train tracks

```

```

622 turnLeftBack(21, 30) #23 25 23 15 16 35 85
623 #drives foward to train tracks
624 drive(150,0,45)
625 wait_for_seconds(0.2) #wait()
626 findBlack(25,30,rightSensor)
627 #drive(225, 0, 45) # we used this before we used find black ^^#500-before turn 550 450 400 350
628 #turns left to aligns itself a little more
629 #wait()
630 turnLeftForward(20, 20) #14 16 10
631 wait_for_seconds(0.2)
632 #drives a little more foward toward the train tracks
633 drive(130, 0, 45) #125 225 35 215 208 230 200 300 175
634 #turns toward the sorting center
635 turnRightForward(79, 30) #80 78 73 76 83 80 70 53 55 85 70 75
636 #wait()
637 wait_for_seconds(0.2) #wait()
638 #rake goes down so can get blocks
639 moveRake('down', 0.8, 55) #0.5
640 wait_for_seconds(0.5)
641 #drives into sorting center and gets blocks
642 driveStraightForSeconds(25, 2)
643 #has attachments lift down the train tracks and the package gets delivered
644 leftAttachment.run_for_seconds(1,-100)
645 moveRakeForDegrees(180,100)
646 wait_for_seconds(0.5) #0.2
647 #rake pickes up blocks
648 moveRake('up', 1, 100)
649 wait_for_seconds(0.2) #wait()
650 #the ramp and the hook goes back up
651 leftAttachment.run_for_seconds(1, 45)
652 #backs out of the sorting center
653 drive(350,0,-45)
654 #turns right to go back home
655 turnRightBack(70, 30) #7072 70 84 74 72 71
656 wait_for_seconds(0.2) #wait()
657 #drives back home
658 #wait()
659 drive(1700, -5, 70) #700 -5
660 #has the robot align to get back into home straight
661 turnRightForward(12, 45) #10 30
662 #turnLeftForward(48, 30)
663 #turnRightForward(48, 30)
664 #gets into home fully
665 drive(1000, 6, 60) #200 400 600
666 #turnRightForward(70, 40)
667 wait_for_seconds(0.2) #wait()
668 moveRake('down',1,80)
669
670 #leftAttachment.run_for_seconds(0.5, -35)
671 #wait()
672 #leftAttachment.run_for_seconds(2.7, -97)
673 #oneDelivery()
674
675 #
676 #
677 #
678 #

```

```
679 # .....
680 # .....
681 # .....
682 # .....
683 # .....
684 # .....
685 # .....
686 # .....
687 # .....
688 #
689
690 def twoPlace():
691     moveRake('down', 1, 100)
692     wait()
693     leftAttachment.start(25)
694     drive(330, 0, 45) #350 300
695     #turn to avoid the bridge
696     turnRightForward(45, 30)
697     leftAttachment.stop()
698     drive(750, 0, 45) #675 655 600 650 700 750 800
699     turnRightForward(26, 30) #27 24 23 22
700     #drive(235, 0, 45) #535 530 700 800 850 800, 850 750 700
701     #look for line in front of bridge
702     findBlack(45, 40, leftSensor)
703     #drive toward cargo deck
704     drive(380, 0, 45) #100 300 320 350 370 365 375 385
705     turnLeftTank(78, 15) #80, 78, 79
706     wait_for_seconds(0.5)
707     #turnLeftForward(80, 30) #73 71 69 65
708     #Load cargo onto balancing deck
709     driveStraightForSeconds(35, 1.25)
710     driveStraightForSeconds(15, .25)
711     moveRakeForDegrees(360, 50)
712     drive(260, 0, -45) #250 270
713     #move the hook down
714     leftAttachment.start(-100)
715     turnRightForward(82, 30) #70 75 78 75 70 80
716     drive(30, 0, -25) #150 55 50 40 35
717     leftAttachment.stop()
718     #drive forward to unload cargo
719     drive(200, 0, 40) #280 300
720     moveRakeForDegrees(-360, 50)
721     drive(200, 0, -45)
722     #turn away from cargo deck
723     turnRightForward(6, 30) #7 8 5
724     #lift up hook
725     leftAttachment.start(100)
726     wait_for_seconds(0.5)
727     leftAttachment.stop()
728     #drive toward train
729     drive(520, 0, 45) #750 500 480 470 490 600 530
730     #big boi turn
731     turnRightTank(65, 30) #80 82 85 83 88 86 84 67 70 75 73 68
732     #back into wall
733     driveMotors.start(0, -50)
734     wait_for_seconds(1.5) #1 2
```

```

735 driveMotors.stop()
736 #drive(590, 0, -50) #600 570
737 drive(100, 0, 45) #200 150 170 180 150
738 turnLeftTank(35, 15)#20 30
739 #drop off blue block
740 moveRakeForDegrees(150, 90) #90 100
741 drive(100, 0, 45) #100 50
742 #turn and release package
743 turnRightTank(100, 15) #80 40 100
744 #back up into helicopter
745 drive(300, 0, -45) #150 100
746 #turnRightTank(15, 15)
747 #turnLeftTank(15, 15)
748 drive(100, 0, 45)
749 #turn so that the hook has room to go down
750 turnLeftTank(45, 15) #80 40 53 68 58 60 40
751 #back up toward helicopter
752 drive(45, 0, -50) #200 100 75 35
753 wait()
754 #push da train
755 leftAttachment.start(-100)
756 driveStraightForSeconds(30, 3)
757 leftAttachment.stop()
758 drive(200, 0, -50)
759
760
761
762 #twoPlace()
763 #or to not to place. That is the question
764
765
766
767 #
768 #
769 #
770 #
771 #
772 #
773 #
774 #
775 #
776 #
777 #
778 #
779 #
780 #
781 def threeMusketeers():
782     #moveRake('up', 1, 55)
783     leftAttachment.start(6)
784     rightAttachment.start(-50)
785     driveStraightForSeconds(62, 2.73)#hits wall 2.5 3.5, 55
786     leftAttachment.stop()
787     rightAttachment.stop()
788     turnRightTank(30,25)#flips switch engine over
789     turnLeftTank(30,25)#turns to wall

```

```

790 wait_for_seconds(0.5) #1
791 driveStraightForSeconds(25,0.5)#run into wall
792 drive(95,0,-45) #85 65 95 75 #55 #backing away from wall
793 turnRightBack(35,42) #25 #45 #15 #42#turn so back is facing home
794 drive(371,0,-45) #new idea 255 285 #275 #270 #260 #240#back up to handle to cargo ship
795 #wait_for_seconds(0.1)
796 #Putting light saber at the right place
797 leftAttachment.run_for_degrees(81, -60) #68 76 81
798 #wait()
799 wait_for_seconds(0.32)
800 #wait()
801 #driving to air lift
802 drive(100, 0, 45)
803 #wait()
804 wait_for_seconds(0.1)
805 #grabs handle and puts it down
806 leftAttachment.start(-100)#-60 -80
807 wait_for_seconds(0.2) #0.1 0.5
808 turnRightBack(45,40)#turns toward the cargo ship 45
809 #wait()
810 leftAttachment.run_for_seconds(1,50)#puts the attachment up
811 turnLeftBack(45,45)#turn towards the cargo
812 #wait()
813 drive(222,0,25) # 200 150
814 moveRake('down',0.9,100) #55 1.5
815 drive(162,0,-50) #142 122 118
816 moveRake('up',0.85,100) #55 1.5
817 drive(653,0,-70) #553 245 250 275 300 400 500 -50
818
819 #rightAttachment.run_for_degrees(54, -60)
820
821
822
823
824
825 threeMusketeers()
826
827 #
828 #
829 #
830 #
831 #
832 #
833 #
834 #
835 #
836 #
837 #
838 #
839 #
840 #
841
842 def acc_avoid():
843     #driveStraightForSeconds(-40,2)
844     driveMotors.start(0, -50)

```

```
845 wait_for_seconds(1.5)#1 2
846 driveMotors.stop()
847 drive(50, 0, 20)
848 rightAttachment.start(-100)
849 turnRightTank(77, 15)
850 rightAttachment.stop()
851 moveRakeForDegrees(-100,100)
852 driveStraightForSeconds(10,2)
853
854
855 def transportationJourney():
856     leftAttachment.run_for_seconds(1.5,-100)#1
857     leftAttachment.start(100)
858     #leftAttachment.run_for_seconds(1.25,100)#3
859     wait_for_seconds(1)
860     leftAttachment.stop
861     drive(440, 0, 50)#390
862     turnLeftForward(3.93,15) #3.35 2.8 2.4 2
863     leftAttachment.run_for_seconds(1,-100)
864     #leftAttachment.run_for_seconds(3,100)
865     leftAttachment.start(100)
866     wait_for_seconds(1)
867     leftAttachment.stop
868     drive(340, 0, 50)#375 275 250
869     turnRightTank(109, 15)
870
871 #transportationJourney()
872
873 def truck():
874     drive(458, 0, 50) #575 390 455 460
875     turnRightForward(80, 20) #82 35
876     drive(400, 0, 30)
877     moveRake('up', 0.5, 50)
878     drive(100, 0, -50) #500 100
879     drive(100,0,-40)
880     turnLeftBack(17, 20) #40
881     drive(869, 0, 30) #669 before took out transpo jounrey 169
882     turnRightTank(115,15)
883
884 def rakeSetup():
885     wait()
886     moveRake('down',0.5,100)
887     wait()
888
889 def fourKablam():
890     rakeSetup()
891     truck()
892     acc_avoid()
893
894 #fourKablam()
895
896 def masterProgram():
897     #setup
898     runNumber = ['4','3','2','1']
899     index = 3
900     hub.left_button.was_pressed()
901     hub.right_button.was_pressed()
```



```
902 #execution
903 while True:
904     hub.light_matrix.write(runNumber[index%4])
905     if hub.right_button.was_pressed():
906         index += 1
907     if hub.left_button.was_pressed():
908         if index%4 == 3:
909             oneDelivery()
910             index += 3
911         elif index%4 == 2:
912             twoPlace()
913             index += 3
914         elif index%4 == 1:
915             threeMusketees()
916             index += 3
917         elif index%4 == 0:
918             fourKablam()
919             index += 3
920     hub.left_button.was_pressed
921     wait_for_seconds(0.1)
922
923 #masterProgram()
```