**capstoneProject\_L3**

from vexcode import \*

def newGrid():

if gridCounter != 4:

drivetrain.turn\_for(RIGHT, 180, DEGREES)

while not distance.get\_distance(MM) < 75:

wait(5,MSEC)

drivetrain.drive(FORWARD)

drivetrain.stop()

drivetrain.turn\_for(LEFT, 90, DEGREES)

drivetrain.drive\_for(FORWARD, 400, MM)

drivetrain.turn\_for(LEFT, 90, DEGREES)

else:

drivetrain.stop()

def capstone3():

global total

global result

global resultD

global true

global false

global lineCounter

global lineCounterD

global gridCounter

true = 1

false = 0

gridCounter = 0

while gridCounter < 5:

drivetrain.set\_drive\_velocity(100, PERCENT)

drivetrain.set\_turn\_velocity(100, PERCENT)

total = []

result = []

lineCounter = 0

lineCounterD = 1

resultD = 0

drivetrain.drive(FORWARD)

while not distance.get\_distance(MM) < 75:

wait(5,MSEC)

if not down\_eye.detect(NONE):

if down\_eye.detect(GREEN):

total.append(true)

lineCounter = len(total)

brain.print(lineCounter,". Line: ",true)

brain.new\_line()

resultD += lineCounterD

lineCounterD \*= 2

elif down\_eye.detect(BLUE):

total.append(false)

lineCounter = len(total)

brain.print(lineCounter,". Line: ",false)

brain.new\_line()

lineCounterD \*= 2

while not down\_eye.detect(NONE):

wait(5,MSEC)

for x in total[::-1]:

result.append(x)

brain.print(gridCounter+1,". Column's Binary Result: ")

for x in result:

brain.print(result[x])

brain.new\_line()

brain.print(gridCounter+1,". Column's Decimal Result: ",resultD)

brain.new\_line()

newGrid()

gridCounter += 1

def main():

capstone3()

stop\_project()

vr\_thread(main())