# -\*- coding: utf-8 -\*-

##:::::::[ python module for Scribbler 2mms ]:::::::::::::::::::::::::::::::::

## File: robot.py v 1.4

## In which FORWARD has been duplicated as MOVE\_FORWARD and

## a new function named ACCELERATE has been defined. Also the

## calculation of the arguments v, v0 and a is made more explicit.

## Once those values have been calculated, they are passed to

## a new function, APPEND\_MOVE\_DISTANCE\_COMMAND.

##

## Bay School Python Module Derived by

## Katie Partington and Richard Piccioni

## from

## Scribbler II Kinematic GUI

## (c) Copyright 2012 and 2013 Matt Greenwolfe

## See end of file for terms of use.

##

##

## This python module contains functions needed

## to program precision motion of the Scribbler 2 and 3

## robots.

## Dependencies:

## Parallax USB Drivers:

## download from: https://www.parallax.com/downloads/parallax-usb-driver-installer

##

## Python 3.x: See www.python.org for installation instructions

## Download 32-bit version

## During installation, add to PATH

##

## Parallax propellent (sic) dynamic link library (Propellent.dll)

## download from: http://www.parallax.com/PropellerDownloads

##

## Propeller FloatMath.spin library

## download from: http://www.parallax.com/PropellerDownloads

##

## Place S2mms.spin, propellent.dll, and FloatMath.spin in

## the same folder as this file.

##=======[ Introduction ]=========================================================

## s2mms.spin provides low-level motor drivers and higher level spin routines

## that move the S2 Robot based on step function velocity and acceleration

## profiles. Accuracy is close to 1 mm/s or 1 mm/s^2.

##

## This Python module provides the functions needed to specify a series of

## motions, write a spin program, and send that program to the robot.

## Most of the functions defined here ultimately call Matt's S2.MOVE\_DISTANCE\_MMS,

## which has the signature (d, v0, a).

##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

from \_\_future\_\_ import division, print\_function

import os

import sys

import ctypes

commands = []

speed\_limit = 18.00 # cm/s

def append\_move\_distance\_command(d, v0, v, a):

if abs(v0) > speed\_limit or abs(v) > speed\_limit:

print("Oops! Speed limit exceeded.")

return

global commands # Note to CS students: Don't use global variables in complex programs.

# We are using one here to simplify the argument list that IDLE displays

# to physics students when they edit their code.

commands += ["s2.move\_distance\_mms(" + str(d)+ ", " + str(v0) + ", " + str(a) + ")\n"]

def speed\_up\_to(final\_speed, time\_interval):

""" Uniformly accelerate the robot from rest """

v0 = 0.00

d = final\_speed \* time\_interval / 2

a = final\_speed/time\_interval

v = final\_speed

append\_move\_distance\_command(d, v0, v, a)

def move\_forward(distance, time\_interval):

d = distance

v0 = distance / time\_interval

a = 0.00

v = v0

append\_move\_distance\_command(d, v0, v, a)

def move\_backward(distance, time\_interval):

d = distance

v0 = - distance / time\_interval

a = 0.00

v = v0

append\_move\_distance\_command(d, v0, v, a)

def cruise\_at(cruising\_speed, time\_interval):

d = cruising\_speed \* time\_interval

v0 = cruising\_speed

a = 0.00

v = v0

append\_move\_distance\_command(d, v0, v, a)

def stop\_from(initial\_speed, time\_interval):

d = initial\_speed \* time\_interval/2

v0 = initial\_speed

a = -initial\_speed/time\_interval

v = v0

append\_move\_distance\_command(d, v0, v, a)

def accel(initial\_speed, acceleration, time\_interval):

v0 = initial\_speed

Dt = time\_interval

a = acceleration

d = v0 + a \* Dt \* Dt / 2

v = v0 + a \* Dt

append\_move\_distance\_command(d, v0, v, a)

def accelerate(initial\_speed, final\_speed, time\_interval):

v0 = initial\_speed

Dt = time\_interval

d = time\_interval \* (initial\_speed + final\_speed)/2

a = (final\_speed - initial\_speed)/time\_interval

v = v0 + a \* Dt

append\_move\_distance\_command(d, v0, v, a)

def pause\_for(time\_interval):

global commands

commands += ["s2.run\_motors\_mms(0,0,0,0.00,0.00, " +

str(int(time\_interval\*1000)) + ")\n"]

def turn\_left(degrees\_ccw):

global commands

commands += ["s2.turn\_mms(" + str(degrees\_ccw)+ ")\n"]

def send\_command\_list(list\_name = commands):

print("Writing spin file . . . ")

spinfile = "move\_s2mms.spin"

try:

ctype\_spinfile = ctypes.c\_char\_p(spinfile)

except TypeError:

ctype\_spinfile = ctypes.c\_char\_p(spinfile.encode('utf-8'))

spin\_code = '''CON

\_clkmode = xtal1 + pll16x

\_xinfreq = 5\_000\_000

OBJ

s2 : "s2mms"

PUB start

s2.start\_motors

repeat

waitcnt(clkfreq + cnt)

waitpne(|< s2#BUTTON, |< s2#BUTTON,0)

'''

for command in commands:

spin\_code += (" " + command)

with open(spinfile,"w") as ms2:

ms2.write(spin\_code)

ms2.close()

path = os.path.abspath(os.path.dirname(sys.argv[0])) #points to curr working dir

prop = ctypes.cdll.LoadLibrary(path + "\Propellent.dll")

prop.InitPropellent(None)

try:

libdir = ctypes.c\_char\_p(os.path.realpath(path))

except TypeError:

libdir = ctypes.c\_char\_p(os.path.realpath(path).encode('utf-8'))

prop.SetLibraryPath(libdir)

prop.CompileSource(ctype\_spinfile,True)

# prop.DownloadToPropeller(0,1) #store in RAM only?

prop.DownloadToPropeller(0,3) #store in RAM and EEPROM

prop.FinalizePropellent

# legacy translations

left = turn\_left

forward = move\_forward

##=======[ License ]===========================================================

##

##┌──────────────────────────────────────────────────────────────────────────────────────┐

##│ TERMS OF USE: Software License │

##├──────────────────────────────────────────────────────────────────────────────────────┤

##│The purchase of one copy of S2mmsKinematicGUI and it's dependent files S2Curve.py, │

##│S2graph.py, S2Segment.py, S2StatusBar.py, S2ToolBar.py, S2VecAdd.py and s2mms.spin│

##│entitles you to install it on every computer in your school or, for │

##│post-secondary institutions, department. Installation to local machines over a network│

##│is allowed. Purchasers are also permitted to distribute these programs to their │

##│students and instructors for home use. The license is limited to a single campus if │

##│your institution has multiple campuses. │

##│ │

##│The above copyright notice and this permission notice shall be included in all copies │

##│or substantial portions of the Software. │

##│ │

##│THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, │

##│INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A │

##│PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT │

##│HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF │

##│CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE │

##│OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. │

##└──────────────────────────────────────────────────────────────────────────────────────┘

##