## Resource R16 C function Sramps for position path planning

The following C function Sramps can be used to construct position commands that smoothly transition from one to another position over a period of time. It is used in Lab Exercise 08 to define position commands. A file containing this function, called Sramps.c, can be found at ricopic.one/embedded_computing/source/Sramps.c.

```
/*
* Sramps.c
*
*  Created on: Mar 18, 2016
*      Author: garbini
*/
#include "math.h"

typedef struct {
  double xfa; double v; double a; double d;
} seg;

int Sramps(
  seg *segs,
  int nseg,
  int *iseg,
  int *itime,
  double T,
  double *xa
){
// Computes the next position, *xa,
// of a uniform sampled position profile.
// The profile is composed of an array
// of segments (type: seg)
// Each segment consists of:
//      xfa:    final position
//      v:      maximum velocity
//      a:      maximum acceleration
//      d:      dwell time at the final position
// Called from a loop, the profile proceeds from
// the current position,
// through each segment in turn, and then repeats.
// Inputs:
//  seg *segs:  - segments array
//  int nseg:   - number of segments in the profile
//  int *iseg:  - variable hold segment index
//  int *itime  - time index within a segment
//  (= -1 at segment beginning)
```

```
//  double T:   - time increment
// Outputs:
//  double *xa: - next position in profile
// Returns:
//  n - number of samples in the profile,
//  0 otherwise
//
// Call with *itime = -1, *iseg = -1, outside the loop
// to initialize.

double t, t1=0, t2=1, tf=1, tramp;
double x1=1, xramp, xfr=1, xr, d;
static double x0, dir;
static int ntot;
double vmax=1, amax=1;
int n;

if (*itime==-1) {
  (*iseg)++;
  if(*iseg==nseg) {
    *iseg=0;
    ntot = 0;
  }
  *itime=0;
  x0=*xa;
}
vmax=segs[*iseg].v;
amax=segs[*iseg].a;
d=segs[*iseg].d;
xfr=segs[*iseg].xfa-x0;
dir=1.0;
if(xfr<0){
  dir=-1.;
  xfr=-xfr;
}
t1 = vmax/amax;
x1 = 1./2.*amax*t1*t1;
if (x1<xfr/2) {
  xramp = xfr-2.*x1;
  tramp = xramp/vmax;
  t2 = t1+tramp;
  tf = t2+t1;
} else {
  x1 = xfr/2;
  t1 = sqrt(2*x1/amax);
  t2 = t1;
  tf = 2.*t1;
}
n = trunc((tf+d)/T)+1;
```

```
t = *itime*T;
if(t<t1) {
  xr = 1./2.*amax*t*t;
} else if (t>=t1 && t<t2) {
  xr = x1+vmax*(t-t1);
} else if (t>=t2 && t<tf) {
  xr = xfr-1./2.*amax*(tf-t)*(tf-t);
} else {
  xr = xfr;
}
*xa=x0+dir*xr;
(*itime)++;
if(*itime==n+1) {
  ntot = ntot + *itime - 1;
  *itime=-1;
  if(*iseg==nseg-1) {
    return ntot;
  }
}
return 0;
}
```