Krmilno jedro

Boštjan Šuhel
TŠC Maribor - School Center for Technical Sciences
Zolajeva ulica 12, 2000 Maribor
bostjan.suhel@gmail.com

Steering core

Control program in multi-tasking operating system and absolute time synchronization. Control core objects are arbitrarily added, controlled via high-speed communications, and subject to time shaking. Control core objects must have a linear time complexity, be executed quickly, and must not fluctuate during execution. When describing, we limit ourselves to the limitations of multiple task operating systems and do not anticipate bypasses, such as running in a reserved kernel or running programs outside the program assignor. A pulse class is described that describes methods, properties, and events. Real python leads us through working examples to software primitives suitable for teaching or development.

Kratek pregled prispevka

Krmilni program v več opravilnem operacijskem sistemu in absolutno časovno sinhronizacijo. Objekti krmilnega jedra se poljubno dodajajo, so nadzirani preko hitrih komunikacij in so podvrženi časovnemu tresenju. Objekti krmilnega jedra morajo imeti linearno časovno kompleksnost, se morajo hitro izvajat in ne smejo nihati v času izvajanja. Pri opisovanju se omejimo na omejitve več opravilnega operacijskega sistema in ne predvidimo obvodov, kot so delovanje v rezerviranem jedru, ali poganjanje programov izven dodeljevalca programov. Opisan je razred pulz, ki opisuje metode, lastnosti in dogodke. Resnični python nas preko delujočih primerov pripelje do programskih primitiv, primernih za poučevanje ali razvoj.

1 Python

Python[2] je močan in hiter, deluje povsod, je prijazen in enostaven za učenje, je odprt in se odlično vklopi v druge tehnologije. Lahko si ga predstavljamo kot kocko v sestavljanki, ki ji rečemo projekt. Python-uu rečemo jezik dogodkov. Po delovanju je tolmač ali prevajalnik, kar nam ustreza. Dodana tehnologija Cython[3] poskrbi za avtomatsko prevajanje knjižnic v C in nato v preveden program. Za pisanje nam ni potrebno poznati dva jezika.

2 Krmilno jedro

Slovenska izdaja[1] je zbirka vaj in projektov. Krmilno jedro je telo zanke while konec(t4):. Predstavljeni program krmili robota kot se vidi na posnetku[4].

```
#<a href="php.php">Python</a> Robot robot.py
#-*- coding: utf-8 -*-
import sys;sys.path.insert(0, '/var/www/html/python')
from python import *

pulz1=pulz();
naklon=0;nagib=0;izris_nastavi(0.2)
while konec(t4):#konec programa
  if tipalo:
     pulz1.premakni(True,0.01,0.2)
     naklon=tipalo[1]
     if naklon<5 and naklon>-5:naklon=0
     nagib=tipalo[2]
     if nagib<5 and nagib>-5:nagib=0
     tipalo.clear()
  if start():
    izhod(m1,pulz1.pwm((naklon+nagib)/1000,0.02))
    izhod(m2,pulz1.pwm((-naklon-nagib)/1000,0.02))
    izhod(m3,pulz1.pwm((naklon-nagib)/1000,0.02))
    izhod(m4,pulz1.pwm((-naklon+nagib)/1000,0.02))
  else:
    izhod(m1,0.0);izhod(m2,0.0)
    izhod(m3,0.0);izhod(m4,0.0)
  izris((naklon+nagib)+50,(naklon-nagib)+50,0)
```

Slika 1: Krmilno jedro.

Telo stavka if tipalo: se izvede ko prispe nova vrednost tipal. Nastavi se vrednost nagib in naklon. Poskrbimo tudi za zelo male vrednosti, da se v primeru , ko držimo telefon vodoravno robot ustavi.

Telo stavka if start(): se izvaja, ko imamo robot v stanju Start. Telo z osnovnimi primitivami opisuje mehanski model našega robota.

Rabimo 4-i primitive izhod(m1,pulz1.pwm((naklon+nagib)/1000,0.02)). Za naša motorja rabimo dva H-mostična krmilnika, 2A, 12V. Izhod m1 do m4 so pini, ki krmilijo vhode teh krmilnikov. Pulz1.pwm je izvedba objekta pwm, ki določi frekvenco in pulzno širino, glede na spremenljivki naklon in nagib. Gre za enostavno opisovanje robotskih mehanskih modelov, kar je ključno za razumevanje.

izris((naklon+nagib)+50,(naklon-nagib)+50,0) je zapis analognih vrednosti v tabelo, ki jo kasneje lahko izrišemo kot trend diagram smeri in hitrosti vrtenja koles. Vrednosti se vpisuje na 0,2s.

3 Knjižnica python.py

```
def konec(p=0):#glavna while zanka
  global konec_pin,konec1,zanka_prvic,cikli,queue12345
  if zanka_prvic==True:
    zanka_prvic=False
    t=threading.Thread(target=UDP_server);t.start()
  konec_pin=p
  cikli = cikli + 1
  time.sleep(0.00005)
  if p==0:
    return not konec1
  else:
    s=not vhod(p)
    if s==False:konec1=True
    return s and (not konec1)
```

Slika 2: konec().

Krmilno jedro nadzorujejo programi iz ozadja. Funkcija konec() vrne True dokler ne pritisnemo gumba Konec in je pin ki ga podamo kot parameter 0.

```
def UDP_server():
  global konec1,pavza,g1s,g2s,g3s,g4s,cikli,staricas,wlanip4,ethip4
  global azimut,naklon,zasuk
  server = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
  try:
    #server.bind(('127.0.0.1', 20001))
    server.bind(('', 20001))
    while not konec1:
      server.settimeout(0.1)
      try:
        azimut=1.2
        s = server.recvfrom(1024)
        sprejem=json.loads(s[0].decode())
        ukaz=sprejem.get("ukaz")
        if ukaz=="tipalo":

oddaja=json.dumps({"ukaz":ukaz,"r1234":str(int(vhod(r1)))+str(int(vhod(r2)))+str(int(vhod(r3)))+str(int(vhod(r4))),"m1234":str(int(vhod(m1)))+str(int(vhod(m2)))+str(int(vhod(m3)))+str(int(vhod(m4))),"t1234":str(int(vhod(t1)))+str(int(vhod(t2)))+str(int(vhod(t3)))+str(int(vhod(t4))),"l1234":str(int(vhod(l1)))+str(int(vhod(l2)))+str(int(vhod(l3)))+str(int(vhod(l4))))
          tipalo.append(sprejem.get("azimut"))
          tipalo.append(sprejem.get("naklon"))
          tipalo.append(sprejem.get("nagib"))
        elif ukaz=='g':
          parameter=sprejem.get("parameter")
          if parameter=='1':g1s=True
          if parameter=='2':g2s=True
          if parameter=='3':g3s=True
          if parameter=='4':g4s=True
```

Slika 3: UDP_server()

Ob prvem delovanju se požene nit s programom UDP_server. UDP_server sprejema UDP telegrame, nastavi lastnosti in vrne UDP odgovor.

4 Integralno razvojno orodje za python

Slika 4: Integralno razvojno orodje za python.

Integralno razvojno orodje [5] za python ima gumbe Status, Start, Stop, Konec, g1, g2, g3 in g4.

Status izpiše status programa, zanimiv je Cikel, ki pove koliko časa se izvaja ena ponovitev jedra.

Start in Stop ustavita ali poženeta izvajanje različne dele programske kode v krmilnem jedru. Če v našem primeru pritisnemo Gumb Stop se robot ustavi.

Konec pomeni konec izvajanja krmilnega jedra.

G1, g2, g3 in g3 so gumbi, ki jih lahko porabimo za posebne ukaze jedru.

Vpisno polje z IP številko 127.0.0.1 je namenjeno ukazovanju drugim jedrom v lokalni mreži. 127.0.0.1 pomeni moj IP.

5 Knjižnica UDP_ukaz.py

```
#Python Uvoz udp_ukaz.py
import socket,sys,json
uporabnik = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
parametrov=len(sys.argv)
if parametrov==3:
  oddaja=json.dumps({"ukaz":sys.argv[2]})
  uporabnik.sendto(oddaja.encode(), (sys.argv[1], 20001))
  uporabnik.settimeout(1)
  try:
    s = uporabnik.recvfrom(1024)
    sprejem=json.loads(s[0].decode())
    ukaz=sprejem.get('ukaz')
    if ukaz=='Status':
      print("Naprava:{3:.16s}\nProgram:{2:.16s}\nEth:   {4:.16s}\nWlan:   {5:.16s}\nStatus: {1:.10s}\nCikel: {0:.6s}ms".format(\
        sprejem.get('cikel'),sprejem.get('status'),sprejem.get('program'),sprejem.get('naprava'),\
        sprejem.get('ethip4').rstrip("\n"),sprejem.get('wlanip4').rstrip("\n")))
    elif ukaz=='izris':
      zapisi=sprejem.get('zapisi');zapisii=sprejem.get('zapisii');
      print("\nbaza={0:.6s};perioda={1:.6s};kazalec={2:.6s};bool={3:.6s};\n"\
      .format(str(zapisii[0]),str(zapisii[0]*400),str(int(zapisii[1]/3)),str(zapisii[2])))
      print('zapisi={0:.1500s};'.format(str(zapisi)));
    elif ukaz=='izris1':
      zapisi=sprejem.get('zapisi');
      print('zapisi1={0:.1500s};'.format(str(zapisi)));
    elif ukaz=='izris2':
      zapisi=sprejem.get('zapisi');
      print('zapisi2={0:.1500s};'.format(str(zapisi)));
    elif ukaz=='izris3':
      zapisi=sprejem.get('zapisi');
      print('zapisi3={0:.1500s};'.format(str(zapisi)));
    elif ukaz=='izris4':
```

Spletni program uporablja UDP_ukaz-py za pošiljanje ukaza jedru(torej UDP strežniku) in čaka na odgovor. Vse knjižnice si lahko pogledate v Python→Uvoz. Vendar s temi ukazi nebi mogli voziti robota. Pametni telefon[6] je poln tipal. Izkoristimo jih 😊.

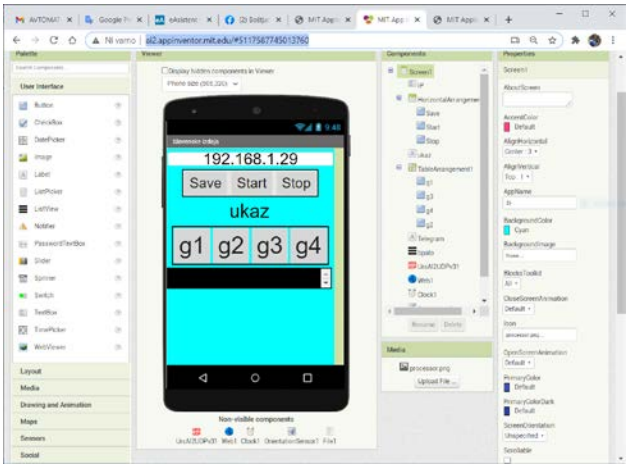
6 Android aplikacija Si

Slika 5: Aplikacija Si.

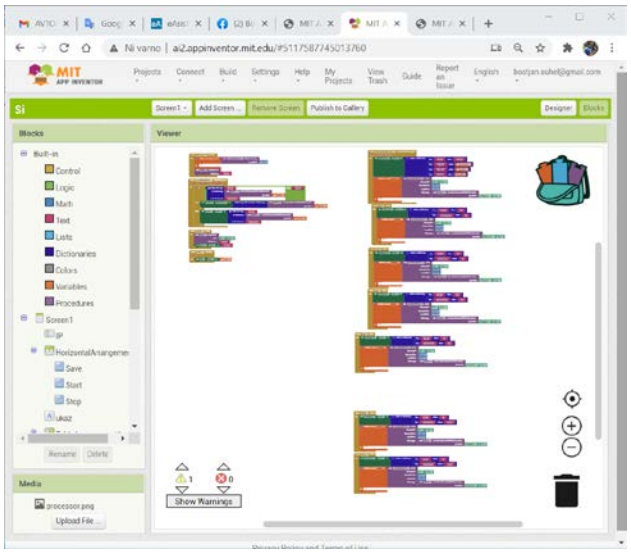
Android[7] aplikacija Si podpira že opisane funkcije gumbov Start, Stop, g1, g2, g3 in g4. Dodatno je podprto pošiljanje podatkov orientacije telefona in sprejemanje stanja pinov na robotu. Pošiljanje podatka o orientaciji se pošilja cca 10 krat na sekundo. Aplikacija sprejme podatke iz UDP-serverja kot odgovor in primer, kako pošiljamo povratne informacije iz robota.

7 AppInventor

MIT AppInventor[8], je javno dostopno spletno mesto, namenjeno razvoju android aplikacij. Veliko primerov je na slovenski izdaji Orodja→AppInventor[9]. Tu najdete projekt(si.aia) in aplikacijo za namestitev (si.apk).



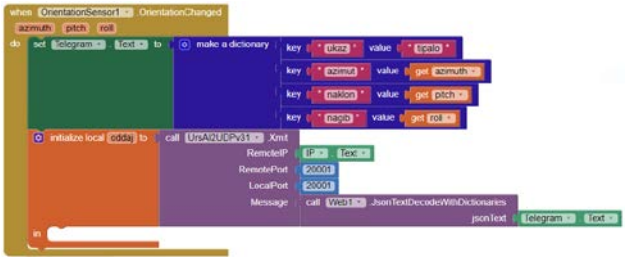
Slika 6: Zaslon.



Slika 7: Bloki.

Aplikacija izvaja podobno funkcijo kot uporabniški program UDP_ukazy.py. Na Zaslonu imamo dodano tipalo orientacije in modul za UDP komunikacijo.

8 Dogodek OrientationSensor



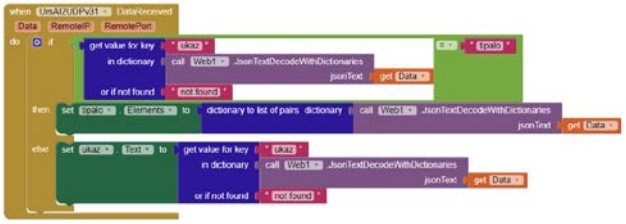
Tipalo OrientationSensor ima dogodek OrientationChanged. Ob spremembi podatkov se najprej pripravi telegram(text). Po standardih se vpišejo elementi slovarja. Vpišemo ukaz, azimut, naklon in nagib. Ukaz je »tipalo«.

9 Odziv UDP_server()

```
        if ukaz=='tipalo':
oddaja=json.dumps({"ukaz":ukaz,"r1234":str(int(vhod(r1)))+str(int(vhod(r2)))+str(int(vhod(r3)))+str(int(vhod(r4))
),"m1234":str(int(vhod(m1)))+str(int(vhod(m2)))+str(int(vhod(m3)))+str(int(vhod(m4))),"t1234":str(int(vhod(t1)))+
str(int(vhod(t2)))+str(int(vhod(t3)))+str(int(vhod(t4))),"l1234":str(int(vhod(l1)))+str(int(vhod(l2)))+str(int(vh
od(l3)))+str(int(vhod(l4)))})
          tipalo.append(senzorji.get("azimut"))
          tipalo.append(senzorji.get("naklon"))
          tipalo.append(senzorji.get("nagib"))
        elif ukaz=='g':
          parameter=senzorji.get("parameter")
          if parameter=='1':g1s=True
          if parameter=='2':g2s=True
          if parameter=='3':g3s=True
          if parameter=='4':g4s=True
          oddaja=json.dumps({"ukaz":ukaz+parameter})
        elif ukaz=='s':
          parameter=senzorji.get("parameter")
          if parameter=='Start':pavza=True
          if parameter=='Stop':pavza=False
          if parameter=='Konec':konec1=True;pavza=True;
          oddaja=json.dumps({"ukaz":parameter})
        elif ukaz=='Status':
          trenutnicas=time.time();cas=trenutnicas-staricas;staricas=trenutnicas
          naprava = socket.gethostname()
          if start():status="Start"
          else: status="Stop"
          cikel=cas/cikli*1000;cikli=0;
```

Pri ukazu »tipalo« se prebere azimut, naklon in nagib. V odgovor se doda digitalne vrednosti pinov.

10 Dogodek DataRecived



Dogodek Datareceived prebere prejeti UDP telegram, ki ga pošlje UDP_server. Zanimajo nas samo telegrami z ukazom tipalo. UDP_server vrne tudi vrednost pinov, ki jih izpišemo na zaslon.

11 Vizija

Projekti so dodatek vajam, kjer se ukvarjamo z vsebinami in pedagoškimi dinamikami. Sledimo trendom uporabe delujočih vaj in projektov. Boljši učenci začnejo vaje spreminjati in kombinirati.

Projekti so dober primer prepletenosti tehnologij in tipičnih problemov, ki jih moramo rešiti. Združeno s standardnimi strežniki in komunikacijami dobimo učinkovito orodje za poučevanje.

12 Literatura

[1] http://si.raspberryip.com/
[2] https://www.python.org/
[3] https://cython.org/
[4] https://www.youtube.com/watch?v=XVWuBPU3RL4
[5] https://sl.wikipedia.org/wiki/Integrirano_razvojno_okolje
[6] https://sl.wikipedia.org/wiki/Pametni_telefon
[7] https://sl.wikipedia.org/wiki/Android_(operacijski_sistem)
[8] https://appinventor.mit.edu/
[9] http://si.raspberryip.com/appinventor/