# **Příloha 1:** Kód řízení v jazyce Python pro křižovatku Lodžská x Mazurská

```
from AAPI import *

# Definice proměnných

NODE_ID = 414  # ID uzlu

# souřadnice středů kruhů pro detekci chodců
r = 3

DPAX1 = 112.93
DPAY1 = 149.24

DPAX2 = 122.10
DPAY2 = 146.77

DPCX1 = 143.50
DPCY1 = 128.34

DPCHX = 142.10
DPCHY = 122.35

DPDX1 = 132.96
DPDY1 = 101.49

DPDX2 = 124.05
DPDY2 = 103.26

DPHX1 = 140.82
DPHY1 = 116.14

DPIX1 = 103.85
DPIY1 = 108.76

DPIX2 = 112.48
DPIY2 = 106.48

DPFX1 = 93.28
DPFY1 = 136.43

DPFX2 = 91.26
DPFY2 = 127.34

DPGX1 = 141.85
DPGY1 = 142.25

DPGX2 = 133.19
DPGY2 = 143.92
```

```
# ID F - Fází, FP - Mezifází
F1 = 1
F2 = 10
F3 = 20
FP12 = 2 # 2, 3, 4, 5
FP13 = 6 # 6, 7, 8, 9
FP21 = 11 # 11, 12, 13, 14
FP23 = 15 # 15, 16, 17, 18, 19
FP32 = 21 # 21, 22, 23, 24
FP31 = 25 # 25, 26, 27, 28

# ID signálních skupin
PAG = 7
PCH = 8
PDI = 9
PF = 10

# Délky fází:
tF1min = 8
tF1max = 29
tF1bus = 47
tF2min = 5
tF2max = 12
tF2bus = 30
tF3min = 8
tF3max = 13
tF3bus = 30
tCM = 2

# Výzvové detektory
DVA = 831
DVB = 837
DVC = 828
DVD = 836
DVE = 829
DVF = 830

# Prodlužovací detektory
DVAi = 893
DVBi = 894
DVCi = 895
DVDi = 896
DVEi = 898
DVFi = 897

# Detektory BUS
DBA = 1284
DBAi = 1287
DBD = 1285
DBE = 1285
DBF = 1286

# EV detektory
```

```
DEA = 1313
DEB = 1313
DEC = 1306
DED = 1285
DEE = 1285
DEF = 1286


# Funkce výpočtu délky časové mezery
class Mezera:
  def __init__(self):
    self.D_start_time = {}
    self.D_previousPress = {}
    self.D_start_time_bus = {}
    self.D_previousPress_bus = {}

    self.D_start_time[DVAi] = 0
    self.D_start_time[DVBi] = 0
    self.D_start_time[DVCi] = 0
    self.D_start_time[DVDi] = 0
    self.D_start_time[DVEi] = 0
    self.D_start_time[DVFi] = 0

    self.D_previousPress[DVAi] = False
    self.D_previousPress[DVBi] = False
    self.D_previousPress[DVCi] = False
    self.D_previousPress[DVDi] = False
    self.D_previousPress[DVEi] = False
    self.D_previousPress[DVFi] = False

    self.D_start_time_bus[DBA] = 0
    self.D_start_time_bus[DBD] = 0
    self.D_start_time_bus[DBE] = 0
    self.D_start_time_bus[DBF] = 0

    self.D_previousPress_bus[DBA] = False
    self.D_previousPress_bus[DBD] = False
    self.D_previousPress_bus[DBE] = False
    self.D_previousPress_bus[DBF] = False

  def Gap(self,time, D):  # Všechna vozidla
    npress = AKIDetGetPresenceInstantDetectionbyId(D, 0, 0.8) == 0

    if npress and not self.D_previousPress[D]:
      self.D_start_time[D] = time

    self.D_previousPress[D] = npress
    if npress:
      return time - self.D_start_time[D]
    else:
      return 0

  def GapB(self,time, D): # Autobusy
    npress = AKIDetGetPresenceInstantDetectionbyId(D, 1, 0.8) == 0
```

```
      if npress and not self.D_previousPress_bus[D]:
         self.D_start_time_bus[D] = time

      self.D_previousPress_bus[D] = npress
      if npress:
         return time - self.D_start_time_bus[D]
      else:
         return 0

mezera = Mezera()

def AAPILoad():
   AKIPrintString( "AAPILoad" )
   return 0

def AAPIInit():
   AKIPrintString( "AAPIInit" )
   return 0

def AAPISimulationReady():
   AKIPrintString( "AAPISimulationReady" )
   return 0

def AAPIManage(time, timeSta, timeTrans, acycle):
   AKIPrintString( "AAPIManage" )

   # Detekce chodců
   Xp = 0
   Yp = 0
   DPA1 = []
   DPA2 = []
   DPC1 = []
   DPCH = []
   DPD1 = []
   DPD2 = []
   DPF1 = []
   DPF2 = []
   DPG1 = []
   DPG2 = []
   DPH1 = []
   DPI1 = []
   DPI2 = []

   for i in range(1,537):
      p = AKIPedestrianGetInf(i)
      if p.report == 0:
         Xp = (p.position.x)
         Yp = (p.position.y)

         if (Xp - DPAX1)**2 + (Yp - DPAY1)**2 <= r**2 :
            DPA1.append(i)
         if (Xp - DPAX2)**2 + (Yp - DPAY2)**2 <= r**2 :
            DPA2.append(i)
```

```
        if (Xp - DPCX1)**2 + (Yp - DPCY1)**2 <= r**2 :
          DPC1.append(i)
        if (Xp - DPCHX)**2 + (Yp - DPCHY)**2 <= r**2 :
          DPCH.append(i)
        if (Xp - DPDX1)**2 + (Yp - DPDY1)**2 <= r**2 :
          DPD1.append(i)
        if (Xp - DPDX2)**2 + (Yp - DPDY2)**2 <= r**2 :
          DPD2.append(i)
        if (Xp - DPFX1)**2 + (Yp - DPFY1)**2 <= r**2 :
          DPF1.append(i)
        if (Xp - DPFX2)**2 + (Yp - DPFY2)**2 <= r**2 :
          DPF2.append(i)
        if (Xp - DPGX1)**2 + (Yp - DPGY1)**2 <= r**2 :
          DPG1.append(i)
        if (Xp - DPGX2)**2 + (Yp - DPGY2)**2 <= r**2 :
          DPG2.append(i)
        if (Xp - DPHX1)**2 + (Yp - DPHY1)**2 <= r**2 :
          DPH1.append(i)
        if (Xp - DPIX1)**2 + (Yp - DPIY1)**2 <= r**2 :
          DPI1.append(i)
        if (Xp - DPIX2)**2 + (Yp - DPIY2)**2 <= r**2 :
          DPI2.append(i)

      # Logické podmínky - výzva chodců na přechodech
      # - PC, PF nebo PH
    LP1 = (len(DPC1) or len(DPCH) or len(DPH1) or len(DPF1) or len(DPF2)) != 0
      # - PA, PD PG nebo PI
    LP3 = (len(DPA1) or len(DPA2) or len(DPD1) or len(DPD2) or len(DPG1) or len(DPG2) or
len(DPI1) or len(DPI2)) != 0

    # Logické podmínky - preferenční nárok BUS
    LB110 = AKIDetGetPresenceCyclebyId(DBA, 1) == 1 # Preferenční nárok BUS VA před zast.
    LB110i = AKIDetGetPresenceCyclebyId(DBAi, 1) == 1# Preferenční nárok BUS VA za zast.

    LB120 = 0
    LB210 = 0
    b = AKIDetGetInfVehInOverStaticInfVehInstantDetectionbyId(DBD, 0, 1, acycle)
    if b.report == 0:
      if b.centroidDest == 566:
        LB120 = True # Preferenční nárok BUS VD
        #    return vBD
      elif b.centroidDest == 558:
        LB210 = True #Preferenční nárok BUS VE
    LB310 = AKIDetGetPresenceCyclebyId(DBF, 1) == 1

    # Logické podmínky - výzva vozidel na detektorech
    LV11 = AKIDetGetPresenceCyclebyId(DVA, 0) == 1 or AKIDetGetPresenceCyclebyId(DVD, 0)
== 1
    LV21 = AKIDetGetPresenceCyclebyId(DVB, 0) == 1 or AKIDetGetPresenceCyclebyId(DVE, 0)
== 1
    LV31 = AKIDetGetPresenceCyclebyId(DVE, 0) == 1 or AKIDetGetPresenceCyclebyId(DVF, 0)
== 1

    # Logické podmínky - ukončení prodlužování vozidel
```

```
LV12 = (mezera.Gap(time,DVAi) >= tCM and mezera.Gap(time,DVDi)) >= tCM
LV22 = (mezera.Gap(time,DVBi) >= tCM and mezera.Gap(time,DVEi)) >= tCM
LV32 = (mezera.Gap(time,DVCi) >= tCM and mezera.Gap(time,DVFi)) >= tCM

# Logické podmínky - preferenční nárok EV
LE11 = 0
LE21 = 0
eab = AKIDetGetInfVehInOverStaticInfVehInstantDetectionbyId(DEA, 0, 6, acycle)
if eab.report == 0:
   if eab.centroidDest == (555 or 558):
      LE11 = True # Preferenční nárok EV VA
   # if eab.centroidDest == 558:
   #    LE11 = True
   elif eab.centroidDest == 563:
      LE21 = True #Preferenční nárok EV VB

LE12 = 0
LE22 = 0
ede = AKIDetGetInfVehInOverStaticInfVehInstantDetectionbyId(DED, 0, 6, acycle)
if ede.report == 0:
   if ede.centroidDest == (566 or 563):
      LE12 = True
   # if ead.centroidDest == 563:
   #    LE12 = True # Preferenční nárok EV VD
   elif ede.centroidDest == 558:
      LE22 = True #Preferenční nárok EV VE


LE10 = LE11 and LE12
LE20 = LE21 and LE22
LE31 = AKIDetGetPresenceCyclebyId(DEC, 6) == 1
LE32 = AKIDetGetPresenceCyclebyId(DEF, 6) == 1
LE30 = LE31 and LE32

currentPhase = ECIGetCurrentPhase( NODE_ID )
phaseTime = 0

global previousInterphase
previousPhase = 0
```