Příloha 2: Kód řízení v jazyce Python pro křižovatku Čimická x K Pazderkám

```
from AAPI import *

# Parametry

# ID uzlů
NODE_ID = 588
NODE_ID2 = 582

# Délky fází
N1N = 8
N1S = 10 - N1N
N1K = 23 - N1N
N1KB = 46 - N1N
N1KBX = 50 - N1N
N2N = 4
N2S = 9
N2K = 19
N3N = 6
N3D1 = 10
N3D2 = 17
N3D3 = 25
N4N = 0
N4S = 15
N4K = 39
N4KB = 54
N4KBX = 58

# Délky mezer
passageTime = 0.5
passageTimeB = 4

# Výzvové detektory
DVA = 670
DVB = 671
DVC = 669
DVD = 673
DVE = 672

# Prodlužovací detektory
DVAi = 765
DVBi = 856
DVCi = 768
DVDi = 857
DVEi = 767

#Detektory bus
DBA = 930
```

```
DBB = 931
DBC = 931
DBE = 932
# Detektory EV
DEA = 930
DEB = 931
DEC = 931
DED = 1108
DEE = 932

#Chodecká tlačítka
r = 3

DPAX = -2203.97
DPAY = 4227.89

DPBX1 = -2178.70
DPBY1 = 4238.03

DPBX2 = -2182.28
DPBY2 = 4232.55

DPCX = -2185.40
DPCY = 4226.98

DPDX = -2185.22
DPDY = 4196.27

DPEX = -2207.60
DPEY = 4200.89

DPFX = -2190.47
DPFY = 4230.15

DPGX = -2181.92
DPGY = 4205.08

DPHX = -2202.16
DPHY = 4195.09

DPIX = -2209.23
DPIY = 4223.63

DPAFX = -2196.88
DPAFY = 4228.98

DPCGX = -2183.68
DPCGY = 4216.45

DPDHX = -2194.64
DPDHY = 4195.44

DPEIX = -2208.5
DPEIY = 4214.19
```

```
# ID fází
F1 = 1
F2 = 6
F3 = 9
F4 = 10
F5 = 13

# ID mezifází (fázových přechodů)
FP1_2 = 2
FP1_2k = 4
FP1_3 = 7
FP1_5 = 11
FP2_1 = 14
FP3_2 = 5
FP4_1 = 17
FP4_5 = 22
FP4_5k = 28
FP5_1 = 20

# ID signálních skupin
SB = 1
VB = 1
PB = 2
VD = 3
KA = 10
PG = 13

# Funkce výpočtu délky červené/ zelené
class Citac:
  def __init__(self):
    self.SG_start_time_g = {}
    self.SG_previous_g = {}
    self.SG_start_time_r = {}
    self.SG_previous_r = {}

    self.SG_previous_g[SB] = False
    self.SG_previous_g[PB] = False
    self.SG_previous_g[VD] = False
    self.SG_previous_g[KA] = False
    self.SG_previous_g[PG] = False

    self.SG_start_time_g[SB] = 0
    self.SG_start_time_g[PB] = 0
    self.SG_start_time_g[VD] = 0
    self.SG_start_time_g[KA] = 0
    self.SG_start_time_g[PG] = 0

    self.SG_previous_r[SB] = False
    self.SG_previous_r[PB] = False
    self.SG_previous_r[VD] = False
    self.SG_previous_r[KA] = False
    self.SG_previous_r[PG] = False

    self.SG_start_time_r[SB] = 0
    self.SG_start_time_r[PB] = 0
    self.SG_start_time_r[VD] = 0
    self.SG_start_time_r[KA] = 0
    self.SG_start_time_r[PG] = 0

  def t(self, time, Node, SG):

    g = ECIGetCurrentStateofSignalGroup(Node, SG) == 1

    if g and not self.SG_previous_g[SG]:
      self.SG_start_time_g[SG] = time

    self.SG_previous_g[SG] = g
    if g:
      return time - self.SG_start_time_g[SG]
    else:
      return 0

  def tk(self, time,Node, SG):
    r = ECIGetCurrentStateofSignalGroup(Node, SG) == 0

    if r and not self.SG_previous_r[SG]:
      self.SG_start_time_r[SG] = time

    self.SG_previous_r[SG] = r
    if r:
      return time - self.SG_start_time_r[SG]
    else:
      return 0

citac = Citac()

# 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[DVA] = 0
    self.D_start_time[DVB] = 0
    self.D_start_time[DVC] = 0
    self.D_start_time[DVD] = 0
    self.D_start_time[DVE] = 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[DVA] = False
        self.D_previousPress[DVB] = False
        self.D_previousPress[DVC] = False
        self.D_previousPress[DVD] = False
        self.D_previousPress[DVE] = False

        self.D_start_time_bus[DBA] = 0
        self.D_start_time_bus[DBB] = 0
        self.D_start_time_bus[DBC] = 0
        self.D_start_time_bus[DBE] = 0

        self.D_previousPress_bus[DBA] = False
        self.D_previousPress_bus[DBB] = False
        self.D_previousPress_bus[DBC] = False
        self.D_previousPress_bus[DBE] = False

    def Gap(self,time, D):
        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):
        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" )

    # Funkce:
    global previousPhase
    phaseTime = 0
    currentPhase = ECIGetCurrentPhase( NODE_ID )

    # Logické podmínky - výzva vozidel na detektorech
    PDVA = AKIDetGetPresenceCyclebyId(DVAi, 0) > 0 #L21
    PDVB = AKIDetGetPresenceCyclebyId(DVBi, 0) > 0 #L22
    PDVC = AKIDetGetPresenceCyclebyId(DVCi, 0) > 0
    PDVD = AKIDetGetPresenceCyclebyId(DVDi, 0) > 0
    PDVE = AKIDetGetPresenceCyclebyId(DVEi, 0) > 0

    # Logické podmínky – délka časové mezery
    L15 = mezera.Gap(time,DVBi)>= passageTime and mezera.Gap(time,DVB) >= passageTime
    L16 = mezera.Gap(time,DVCi)>= passageTime and mezera.Gap(time,DVC) >= passageTime
    L17 = mezera.Gap(time,DVEi)>= passageTime and mezera.Gap(time,DVE) >= passageTime
    L25 = mezera.Gap(time,DVDi)>= passageTime and mezera.Gap(time,DVD) >= passageTime
    L45 = mezera.Gap(time,DVAi)>= passageTime and mezera.Gap(time,DVA) >= passageTime

    L20 = PDVA or PDVD #L21 v L22

    # Logické podmínky - výzva chodců na přechodech
    L30  = []
    L33  = []
    L34  = []
    L35  = []
    L50  = []
    ADPC  = []
    ADPE  = []
    ADPI  = []

    Xp = 0
    Yp = 0
    for i in range(1,91):
        p = AKIPedestrianGetInf(i)
        if p.report == 0:
            Xp = (p.position.x)
            Yp = (p.position.y)

            if  (Xp - DPEIX)**2 + (Yp - DPEIY)**2 <= r**2:
               L30.append(i)
            if  (Xp - DPCGX)**2 + (Yp - DPCGY)**2 <= r**2:
               L33.append(i)
            if  (Xp - DPGX)**2 + (Yp - DPGY)**2 <= r**2:
               L34.append(i)
            if  (Xp - DPBX1)**2 + (Yp - DPBY1)**2 <= r**2:
               L35.append(i)
            if  (Xp - DPBX2)**2 + (Yp - DPBY2)**2 <= r**2:
               L50.append(i)
            if  (Xp - DPCX)**2 + (Yp - DPCY)**2 <= r**2:
               ADPC.append(i)
            if  (Xp - DPEX)**2 + (Yp - DPEY)**2 <= r**2:
               ADPE.append(i)
            if  (Xp - DPIX)**2 + (Yp - DPIY)**2 <= r**2:
               ADPI.append(i)

    L31 = (len(L30) or len(L33)) !=0 # Výzva na PE nebo PG
    L32 = (len(ADPC) or len(ADPE) or len(ADPI)) != 0 # Výzva na PC, PE nebo PI
    L36 = (len(L33) or len(L34) or len(L35) or len(L50)) != 0 # Výzva na PC, PG nebo PB

    # Logické podmínky - preferenční nárok EV
    LE10 = 0
    LE11 = 0

    ebc = AKIDetGetInfVehInOverStaticInfVehInstantDetectionbyId(DEA, 0, 6, acycle)
    if ebc.report == 0:
        if ebc.centroidDest == (642 or 412):
            LE11 = True # Preferenční nárok EV na VC
        elif ebc.centroidDest == 645:
            LE10 = True #Preferenční nárok EV na VB

    LE12 = AKIDetGetPresenceCyclebyId(DEE, 6) == 1 # Preferenční nárok EV na VE
    LE40 = AKIDetGetPresenceCyclebyId(DEA, 6) == 1 # Preferenční nárok EV na VA
    LE41 = AKIDetGetPresenceCyclebyId(DED, 6) == 1 # Preferenční nárok EV na VD

    # Preferenční nárok Bus a EV
    # - Bus a EV na VB
    LB100 = AKIDetGetPresenceCyclebyId(DBB, 1)  == 1 and LE10
    # - Bus a EV na VC
    LB110 = AKIDetGetPresenceCyclebyId(DBC, 1)  == 1 and LE11
    # - Bus a EV na VE
    LB120 = AKIDetGetPresenceCyclebyId(DBE, 1)  == 1 and LE12
    # - Bus a EV na VA
    LB400 = AKIDetGetPresenceCyclebyId(DBA, 1)  == 1 and LE40 and LE41

    # Konec prodlužování Bus a EV
    # - Bus a EV na VB
    LB155 = AKIDetGetPresenceCyclebyId(DBB, 1) == 0 and not LE10
    # - Bus na VB před stopčárou
    LB156 = mezera.GapB(time,DBB) >= passageTimeB
    # - Bus a EV na VC
    LB165 = AKIDetGetPresenceCyclebyId(DBC, 1) == 0 and not LE11
    # - Bus na VC před stopčárou
    LB166 = mezera.GapB(time,DBC) >= passageTimeB
    # - Bus a EV na VE
    LB175 = AKIDetGetPresenceCyclebyId(DBE, 1) == 0 and not LE12
    # - Bus na VE před stopčárou
    LB176 = mezera.GapB(time,DBE) >= passageTimeB
    # - Bus a EV na VA
    LB455 = AKIDetGetPresenceCyclebyId(DBA, 1) == 0 and not (LE40 and LE411)
    # - Bus na VA před stopčárou
    LB456 = mezera.GapB(time,DBA) >= passageTimeB
```

```
 # Nastavení délky F3
 T3 = N3N
 if (L30 or L50) and T3 < N3D1 :
   T3 = N3N
 if L31 and T3 < N3D1:
   T3 = N3D1
 if(L32 or L33) and T3 < N3D3:
   T3 = N3D2
 if L34 or L35:
   T3 = N3D3

 # Fáze 1
 if currentPhase == F1:
   phaseTime = time - ECIGetStartingTimePhase( NODE_ID )
   previousPhase = currentPhase

   pod11 = (LB400 and phaseTime >= N1S) and (LB165 and LB166 and LB175 and LB176 and
((LB155 and LB156) or not L36))
   pod12 = (LB165 and LB166 and LB175 and LB176 and ((LB155 and LB156) or not L36)) and
(((L15 or not L36) and L16 and L17))
   pod13 = (LB165 and LB166 and LB175 and LB176 and ((LB155 and LB156) or not L36)) and
phaseTime >= N1K
   pod14 = (LB166 and LB176 and (LB156 or not L36) and phaseTime >= N1KB) or phaseTime
>= N1KBX

   if phaseTime >= N1N:
     if (L20 or L30 or L31 or L32 or L33 or L34 or L35 or L50 or LB400):
       if (pod11 or pod12 or pod13 or pod14):

         if L33 or L34 or L35 or (L50 and (L20 or LB400)):
           ECIChangeDirectPhase(NODE_ID, FP1_3, timeSta,time,acycle)
           ECIChangeDirectPhase(NODE_ID2, FP1_3, timeSta,time,acycle)

         elif L20 or LB400 or L30 or L31 or L32:
           ECIChangeDirectPhase(NODE_ID, FP1_2, timeSta,time,acycle)
           ECIChangeDirectPhase(NODE_ID2, FP1_2, timeSta,time,acycle)
         else :

           ECIChangeDirectPhase(NODE_ID, FP1_5, timeSta,time,acycle)
           ECIChangeDirectPhase(NODE_ID2, FP1_5, timeSta,time,acycle)

     else:
       ECIChangeDirectPhase(NODE_ID, F1, timeSta,time,acycle)
       ECIChangeDirectPhase(NODE_ID2, F1, timeSta,time,acycle)

 # Fáze 2
 if currentPhase == F2:
   phaseTime = time - ECIGetStartingTimePhase( NODE_ID )

   if previousPhase == F1:
     if L36:
       ECIDisableEvents(NODE_ID2)
       ECIChangeSignalGroupState(NODE_ID2, SB, 0, timeSta, time, acycle)
       ECIChangeSignalGroupState(NODE_ID2, PB, 1, timeSta, time, acycle)
```

```
            ECIEnableEvents(NODE_ID2)

          if L30 or L31 or L32 or L33 or L34 or L35 or L50:
            ECIChangeDirectPhase(NODE_ID, F3, timeSta,time,acycle)
            ECIChangeDirectPhase(NODE_ID2, F3, timeSta,time,acycle)

        if previousPhase == F3 :
          ECIDisableEvents(NODE_ID2)
          ECIChangeSignalGroupState(NODE_ID2, SB, 1, timeSta, time, acycle)
          ECIEnableEvents(NODE_ID2)

        previousPhase = currentPhase
        if phaseTime >= N2N:

          if ((((LB110 or LB120) and phaseTime >= N2N) and LB455 and LB456)) and not L35 and
not L50:
            ECIChangeDirectPhase(NODE_ID, FP2_1, timeSta,time,acycle)
            ECIChangeDirectPhase(NODE_ID2, FP2_1, timeSta,time,acycle)

          elif(((L25 and L45) or phaseTime >= N2S) and (L25 or phaseTime >= N2K or LB110 or
LB120)) :

            if L45 and LB455 and LB456 and not L35 and not L50:
              ECIChangeDirectPhase(NODE_ID, FP2_1, timeSta,time,acycle)
              ECIChangeDirectPhase(NODE_ID2, FP2_1, timeSta,time,acycle)

            else:
              ECIChangeDirectPhase(NODE_ID, F4, timeSta,time,acycle)
              ECIChangeDirectPhase(NODE_ID2, F4, timeSta,time,acycle)

          else:
            ECIChangeDirectPhase(NODE_ID, F2, timeSta,time - phaseTime,acycle)
            ECIChangeDirectPhase(NODE_ID2, F2, timeSta,time - phaseTime,acycle)

      # Fáze 3
      if currentPhase == F3:
        previousPhase = currentPhase
        phaseTime = time - ECIGetStartingTimePhase( NODE_ID )
        if L36:
        # if previousPhase == F2:
          ECIDisableEvents(NODE_ID2)
          ECIChangeSignalGroupState(NODE_ID2, SB, 0, timeSta, time, acycle) #id SB = 741
          ECIChangeSignalGroupState(NODE_ID2, PB, 1, timeSta, time, acycle) #id PB = 742
          ECIEnableEvents(NODE_ID2)
        if phaseTime >= T3:
          T3 = N3N

          ECIChangeDirectPhase(NODE_ID, FP3_2, timeSta,time,acycle)
          ECIChangeDirectPhase(NODE_ID2, FP3_2, timeSta,time,acycle)

      # Fáze 4
      if currentPhase == F4:
        phaseTime = time - ECIGetStartingTimePhase( NODE_ID )
```

```
    previousPhase = currentPhase

    if citac.tk(time,NODE_ID, VD) >= 6 and citac.tk(time, NODE_ID, PG) >= 6 and
ECIGetCurrentStateofSignalGroup(NODE_ID, KA) == 0:

      ECIDisableEvents(NODE_ID)
      ECIChangeSignalGroupState(NODE_ID, KA, 1, timeSta, time, acycle)
      ECIEnableEvents(NODE_ID)

    if citac.tk(time,NODE_ID, VD) >= 4 and citac.tk(time,NODE_ID2, PB) >= 2 and (citac.t(time,
NODE_ID2,SB) >= 3 or ECIGetCurrentStateofSignalGroup(NODE_ID2, SB) == 0) and not L35 and
not L50:
      ECIDisableEvents(NODE_ID2)
      ECIChangeSignalGroupState(NODE_ID2, VB, 1, timeSta, time, acycle)
      ECIEnableEvents(NODE_ID2)

    if phaseTime >= N4N:

      pod41 = ((LB110 or LB120) and phaseTime >= N4S) and LB455 and LB456
      pod42 = (LB455 and LB456) and (L45 )
      pod43 = (LB455 and LB456) and phaseTime >= N4K
      pod44 = (LB456 and phaseTime >= N4KB) or phaseTime >= N4KBX

      if pod41 or pod42 or pod43 or pod44:

        if ECIGetCurrentStateofSignalGroup(NODE_ID, KA) == 0  or citac.t(time, NODE_ID, KA)
>= 4:

          if L35 or L50:
            if(ECIGetCurrentStateofSignalGroup(NODE_ID2, SB) == 0 or citac.t(time,NODE_ID2,
SB) >= 5) and (ECIGetCurrentStateofSignalGroup(NODE_ID2, VB) == 0 or citac.t(time,
NODE_ID2, VB) >= 5):

              ECIChangeDirectPhase(NODE_ID, FP4_5, timeSta,time,acycle)
              ECIChangeDirectPhase(NODE_ID2, FP4_5, timeSta,time,acycle)
            else:
              ECIChangeDirectPhase(NODE_ID, F4, timeSta,time - phaseTime ,acycle)
              ECIChangeDirectPhase(NODE_ID2, F4, timeSta,time - phaseTime,acycle)
          else:
            ECIChangeDirectPhase(NODE_ID, FP4_1, timeSta,time,acycle)
            ECIChangeDirectPhase(NODE_ID2, FP4_1, timeSta,time,acycle)
        else:
          ECIChangeDirectPhase(NODE_ID, F4, timeSta,time - phaseTime ,acycle)
          ECIChangeDirectPhase(NODE_ID2, F4, timeSta,time - phaseTime,acycle)

  # Fáze 5
  if currentPhase == F5:
    previousPhase = currentPhase
    phaseTime = time - ECIGetStartingTimePhase( NODE_ID )

    if phaseTime >= 8:
      ECIChangeDirectPhase(NODE_ID, FP5_1, timeSta,time,acycle)
      ECIChangeDirectPhase(NODE_ID2, FP5_1, timeSta,time,acycle)
    # Přepínání mezifází
    if  currentPhase == FP3_2 and previousPhase == F1:
        ECIChangeDirectPhase(NODE_ID, F2, timeSta,time,acycle)
        ECIChangeDirectPhase(NODE_ID2, F2, timeSta,time,acycle)

    if  currentPhase == FP4_1 and previousPhase == F2:
        ECIChangeDirectPhase(NODE_ID, F1, timeSta,time,acycle)
        ECIChangeDirectPhase(NODE_ID2, F1, timeSta,time,acycle)

    if  currentPhase == FP2_1 and previousPhase == F5:
        ECIChangeDirectPhase(NODE_ID, FP5_1, timeSta,time,acycle)
        ECIChangeDirectPhase(NODE_ID2, FP5_1, timeSta,time,acycle)

    if  currentPhase == FP4_5 and previousPhase == F5:
        ECIChangeDirectPhase(NODE_ID, F1, timeSta,time,acycle)
        ECIChangeDirectPhase(NODE_ID2, F1, timeSta,time,acycle)

   if currentPhase == FP5_1 and previousPhase == F4:
        ECIChangeDirectPhase(NODE_ID, F1, timeSta,time,acycle)
        ECIChangeDirectPhase(NODE_ID2, F1, timeSta,time,acycle)

    if  currentPhase == FP4_5k:
        ECIChangeDirectPhase(NODE_ID, F5, timeSta,time,acycle)
        ECIChangeDirectPhase(NODE_ID2, F5, timeSta,time,acycle)

    return 0

def AAPIPostManage(time, timeSta, timeTrans, acycle):
    AKIPrintString( "AAPIPostManage" )
    return 0

def AAPIFinish():
    return 0

def AAPIUnLoad():
    return 0
```