```
from pycm import *
#import gc
import time
import math

print(dir(pycm))
class CVar:
        IsTorqOn                        = False
        nScreenWidth                    = 0
        nScreenHeight                   = 0
        IsResetCommand                  = False
        nID                             = -1
        nOffset                         = 0
        #nCamera_mode                    = 0
        #nCamera_mode_sub                = 0
        nTouch_XY_X0                    = 0
        nTouch_XY_Y0                    = 0
        nTouch_Pos0                     = 0
        nTouch_Pos_X0                   = 0
        nTouch_Pos_Y0                   = 0
        nTouch_XY_X1                    = 0
        nTouch_XY_Y1                    = 0
        nTouch_Pos1                     = 0
        nTouch_Pos_X1                   = 0
        nTouch_Pos_Y1                   = 0
        nButton_0                       = -1
        nButton_1                       = -1
        nBack_Background                = 0
        btn0                            = None
        btn1                            = None
        nPage                           = -1
        nPage_Prev                      = -1

        #fYaw_first                      = 0.0
        #fYaw                            = 0.0
        #fYaw_turn                       = 0.0

        nSpeed                          = 0

        # For Face tracking
        IsTurning_Face                  = True
        nPos_Servo                      = 512
        m_nPwm                          = 0
```

```
        # For Walking
        nKnee                           = 0
        nWalking                        = 0
        nWalking_Prev                   = 0

class CTimer:
        nTimer                          = 0
```

Page 2

```
        IsTimer                         = False

        def Set(self):
                self.IsTimer = True
                self.nTimer = millis()

        def Get(self):
                if self.IsTimer:
                        return millis()-self.nTimer
                return 0

        def Destroy(self):
                self.IsTimer = False

_COLOR_NONE                 = 0
_COLOR_WHITE                = 1
_COLOR_BLACK                = 2
_COLOR_RED                  = 3
_COLOR_GREEN                = 4
_COLOR_BLUE                 = 5
_COLOR_YELLOW = 6
_COLOR_GRAY_LIGHT = 7
_COLOR_GRAY                 = 8
_COLOR_GRAY_DARK = 9

_SHOW_IMAGE                 = 0
_SHOW_TEXT                  = 1
_SHOW_SHAPE                 = 2
_SHOW_NUM                   = 3

_RATIO                      = 1000
_BTN_INDEX                  = 4
#1..5
#[left,top,right,bottom, Unique number (ButtonNumber)]: left, top <0 then 1..5
  position
# Put the coordinates of each button and the unique number you want to put here-step 1/2 [Note: The button number is
  Do not overlap with other buttons.]
_BTN_MENU                   = [ 20 , 40 , 160 , 150 , 1 ]
_BTN_REMOTE_NORMAL = [ 460 , 30 , 520 , 150 , 2 ]
_BTN_REMOTE_FIGHT = [ 660 , 30 , 716 , 150 , 3 ]
```

_BTN_REMOTE_SPECIAL = [846 , 30 , 900 , 150 , 5]

#Menu-Control

_BTN_SUB_OUT = [1 , 1 , 1000 , 700 , 60]

_BTN_SUB_X = [950 , 730 , 990 , 780 , 61]

_BTN_SUB_REMOTE = [100-50 , 850-100 , 100+50 , 850+100 , 62]

_BTN_SUB_STREAM = [250-50 , 850-100 , 250+50 , 850+100 , 64]

_BTN_SUB_FACE_TRACK = [400-50 , 850-100 , 400+50 , 850+100 , 64]

_BTN_SUB_MOTOR = [700-50 , 850-100 , 700+50 , 850+100 , 65]

_BTN_SUB_OFFSET = [850-50 , 850-100 , 850+50 , 850+100 , 66]

Page 3

#Rotate button

_BTN_TURN_L = [20 , 220 , 130 , 330 , 6]

_BTN_TURN_R = [240 , 220 , 350 , 330 , 7]

#-Diagonal

_BTN_MOVE_DG_FL = [40 , 370 , 100 , 490 , 8]

_BTN_MOVE_DG_FR = [260 , 370 , 330 , 490 , 9]

_BTN_MOVE_DG_BL = [40 , 840 , 100 , 930 , 10]

_BTN_MOVE_DG_BR = [260 , 840 , 330 , 930 , 11]

#Go button

_BTN_MOVE_UL = [90 , 500 , 140 , 570 , 12]

_BTN_MOVE_U = [160 , 420 , 210 , 540 , 13]

_BTN_MOVE_UR = [220 , 500 , 280 , 570 , 14]

_BTN_MOVE_L = [60 , 580 , 120 , 700 , 15]

_BTN_MOVE_R = [240 , 580 , 310 , 700 , 16]

_BTN_MOVE_DL = [90 , 730 , 140 , 810 , 17]

_BTN_MOVE_D = [160 , 760 , 210 , 880 , 18]

_BTN_MOVE_DR = [220 , 730 , 280 , 810 , 19]

#Motion speed button

_BTN_SPD = [140 , 590 , 220 , 720 , 20]

#Torque ON/OFF button

_BTN_TORQ = [440 , 520 , 570 , 630 , 21]

#LED change button

_BTN_LED = [440 , 660 , 570 , 770 , 22]

#Wake up button

_BTN_GETUP = [440 , 790 , 570 , 900 , 23]

#Normal mode action button

_BTN_ACT_01 = [630 , 280 , 750 , 380 , 30] # Hand greeting

_BTN_ACT_02 = [820 , 280 , 940 , 380 , 31] #Bending greetings

_BTN_ACT_03 = [630 , 450 , 750 , 550 , 32] # Ceremony

_BTN_ACT_04 = [820 , 450 , 940 , 550 , 33] #provocation

_BTN_ACT_05 = [630 , 630 , 750 , 730 , 34] #

_BTN_ACT_06 = [820 , 630 , 940 , 730 , 35] # Push-ups

_BTN_ACT_07 = [630 , 790 , 750 , 890 , 36] #Applause

_BTN_ACT_08 = [820 , 790 , 940 , 890 , 37] #Side roll

```
#Fight mode action button
_BTN_ACT_09                    = [ 760 , 280 , 810 , 390 , 38 ] #body throw
_BTN_ACT_10                    = [ 760 , 450 , 810 , 560 , 39 ] # Defense

# Motor ID Tag
_MOT_W = 80
_MOT_H = 50

_BTN_ID_01 = [ 350 -_MOT_W, 80 -_MOT_H, 350 +_MOT_W, 80 +_MOT_H, 101 ]
_BTN_ID_02 = [ 300 -_MOT_W, 200 -_MOT_H, 300 +_MOT_W, 200 +_MOT_H, 102 ]
_BTN_ID_03 = [ 650 -_MOT_W, 80 -_MOT_H, 650 +_MOT_W, 80 +_MOT_H, 103 ]
_BTN_ID_04 = [ 700 -_MOT_W, 200 -_MOT_H, 700 +_MOT_W, 200 +_MOT_H, 104 ]
```

Page 4

```
_BTN_ID_05 = [ 400 -_MOT_W, 420 -_MOT_H, 400 +_MOT_W, 420 +_MOT_H, 105 ]
_BTN_ID_06 = [ 400 -_MOT_W, 540 -_MOT_H, 400 +_MOT_W, 540 +_MOT_H, 106 ]
_BTN_ID_07 = [ 600 -_MOT_W, 420 -_MOT_H, 600 +_MOT_W, 420 +_MOT_H, 107 ]
_BTN_ID_08 = [ 600 -_MOT_W, 540 -_MOT_H, 600 +_MOT_W, 540 +_MOT_H, 108 ]
_BTN_ID_09 = [ 400 -_MOT_W, 780 -_MOT_H, 400 +_MOT_W, 780 +_MOT_H, 109 ]
_BTN_ID_10 = [ 400 -_MOT_W, 900 -_MOT_H, 400 +_MOT_W, 900 +_MOT_H, 110 ]
_BTN_ID_11 = [ 600 -_MOT_W, 780 -_MOT_H, 600 +_MOT_W, 780 +_MOT_H, 111 ]
_BTN_ID_12 = [ 600 -_MOT_W, 900 -_MOT_H, 600 +_MOT_W, 900 +_MOT_H, 112 ]
_BTN_ID_13 = [ 250 -_MOT_W, 320 -_MOT_H, 250 +_MOT_W, 320 +_MOT_H, 113 ]
_BTN_ID_14 = [ 750 -_MOT_W, 320 -_MOT_H, 750 +_MOT_W, 320 +_MOT_H, 114 ]
_BTN_ID_15 = [ 400 -_MOT_W, 660 -_MOT_H, 400 +_MOT_W, 660 +_MOT_H, 115 ]
_BTN_ID_16 = [ 600 -_MOT_W, 660 -_MOT_H, 600 +_MOT_W, 660 +_MOT_H, 116 ]
_BTN_ID_17 = [ 500 -_MOT_W, 300 -_MOT_H, 500 +_MOT_W, 300 +_MOT_H, 117 ]

# Reset & Init (Offset)
_BTN_RESET = [ 900-70 , 300-250 , 900+70 , 300 , 82 ]
_BTN_INIT = [ 900-70 , 300 , 900+70 , 300+250 , 83 ]

#Offset setting dialog
_BTN_DIALOG_OK = [ 600-100 , 800-80 , 600+100 , 800+80 , 84 ]
_BTN_DIALOG_CANCEL = [ 400-100 , 800-80 , 400+100 , 800+80 , 85 ]
_BTN_DIALOG_TORQ = [ 600 - 80 , 400-50 , 600 + 80 , 400 + 50 , 86 ]
_BTN_DIALOG_PLUS = [ 640 - 50 , 560-50 , 640 + 50 , 560 + 50 , 87 ]
_BTN_DIALOG_MINUS = [ 360 - 50 , 560-50 , 360 + 50 , 560 + 50 , 88 ]

_PAGE_MENU                     = 0
_PAGE_REMOTE_NORMAL            = 1
_PAGE_REMOTE_FIGHT             = 2
_PAGE_REMOTE_SPECIAL           = 5
_PAGE_STREAM                   = 6
_PAGE_FACE_TRACK               = 7
_PAGE_MOTOR_TEST               = 8
_PAGE_OFFSET                   = 9
_PAGE_OFFSET_DIALOG1           = 10
_PAGE_OFFSET_DIALOG2           = 11
_PAGE_OFFSET_DIALOG_CLOSE = 12
```

```
#Offset applied value in general motion, not motion
#                       0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
aOffset = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ] #, 0, 0, 0]

btnList                         = None
tmrTorqBtn                      = CTimer()
#tmrFace                          = CTimer()

IsPhone                         = False

def ShowPage(nPage):
    #global markerList
    global btnList
    CVar.nPage_Prev = CVar.nPage
```

Page 5

```
    CVar.nPage = nPage

    # Create a page here and assign the button to it. -step 2/2
    if nPage == _PAGE_MENU:
        btnList = [
            _BTN_SUB_OUT,
            _BTN_SUB_X,
            _BTN_SUB_REMOTE,
            _BTN_SUB_STREAM,
            _BTN_SUB_FACE_TRACK,
            _BTN_SUB_MOTOR,
            _BTN_SUB_OFFSET
            ]

        rpi.mode( 0 ) #0: Close, 1: Color, 2: Face Detection, 3: Streaming, 4: Marker, 5: Lane, 6: Emotion, 7: Hand
        smart.write8( 10700 , 0 ) # Smartphone streaming video screen 0: Close, 1: Display

        #Shows the menu bar
        Show_Image( 500 , 850 , 1 )

        Other buttons except #_BTN_SUB_OUT have images.
        img = [ 2 , 3 , 4 , 6 , 7 , 8 ]
        nIndex = 0
        for btn in btnList:
            if (btn != _BTN_SUB_OUT):
                Show_Image(
                    round((btn[ 0 ] + btn[ 2 ]) / 2 ),
                    round((btn[ 1 ] + btn[ 3 ]) / 2 ),
                    img[nIndex]
                )
                nIndex = nIndex + 1

        #Setup_Speed(254, 0)
```

```
        # All motor torque On
        TorqAll( True )
        #All LEDs off (8-byte command is sent directly to address 65)
        DXL( 254 ).write8( 65 , 0 )

        Motion_Ready( 1 )

        #Head to the center...
        OLLO( 1 , const.OLLO_JOINT_POSITION).write( 512 )
        Clear_Text()
elif nPage == _PAGE_REMOTE_NORMAL:
        Clear_All()

        btnList = [
                _BTN_MENU,
                _BTN_REMOTE_NORMAL,
                _BTN_REMOTE_FIGHT,
                _BTN_REMOTE_SPECIAL,
```

Page 6

```
                #Rotate button
                _BTN_TURN_L,
                _BTN_TURN_R,
                #-Diagonal
                _BTN_MOVE_DG_FL,
                _BTN_MOVE_DG_FR,
                _BTN_MOVE_DG_BL,
                _BTN_MOVE_DG_BR,
                #Go button
                _BTN_MOVE_UL,
                _BTN_MOVE_U,
                _BTN_MOVE_UR,
                _BTN_MOVE_L,
                _BTN_MOVE_R,
                _BTN_MOVE_DL,
                _BTN_MOVE_D,
                _BTN_MOVE_DR,
                _BTN_MOVE_DR,
                #Motion speed button
                _BTN_SPD,
                #Torque ON/OFF button
                _BTN_TORQ,
                #LED change button
                _BTN_LED,
                #Wake up button
                _BTN_GETUP,
                #Normal mode action button
                _BTN_ACT_01, #Hand greeting
                _BTN_ACT_02, #Bending greetings
                _BTN_ACT_03, # ceremony
                _BTN_ACT_04, # provocation
```

```
                _BTN_ACT_05, # mulgunamu
                _BTN_ACT_06, #Push-ups
                _BTN_ACT_07, # Applause
                _BTN_ACT_08 #Side roll
                ]
        Show_Background( 1 + CVar.nSpeed)
        Motion_Ready( 1 )
        nMotion_Prev = 0
elif nPage == _PAGE_REMOTE_FIGHT:
        Clear_All()

        btnList = [
                _BTN_MENU,
                _BTN_REMOTE_NORMAL,
                _BTN_REMOTE_FIGHT,
                _BTN_REMOTE_SPECIAL,
                #Rotate button
                _BTN_TURN_L,
                _BTN_TURN_R,
                #-Diagonal
```



```
                _BTN_MOVE_DG_FL,
                _BTN_MOVE_DG_FR,
                _BTN_MOVE_DG_BL,
                _BTN_MOVE_DG_BR,
                #Go button
                _BTN_MOVE_UL,
                _BTN_MOVE_U,
                _BTN_MOVE_UR,
                _BTN_MOVE_L,
                _BTN_MOVE_R,
                _BTN_MOVE_DL,
                _BTN_MOVE_D,
                _BTN_MOVE_DR,
                _BTN_MOVE_DR,
                #Motion speed button
                _BTN_SPD,
                #Torque ON/OFF button
                _BTN_TORQ,
                #LED change button
                _BTN_LED,
                #Wake up button
                _BTN_GETUP,
                #Normal mode action button
                _BTN_ACT_01, #Front left
                _BTN_ACT_02, #right front
                _BTN_ACT_03, #Front left waist straight
                _BTN_ACT_04, #Right front waist straight
                _BTN_ACT_05, #left side
```

```
            _BTN_ACT_06, #right side
            _BTN_ACT_07, #left turn punch
            _BTN_ACT_08, #Right turn punch
            _BTN_ACT_09, #body throw
            _BTN_ACT_10, # Protection
            ]
        Show_Background( 3 + CVar.nSpeed)
        Motion_Ready( 2 )
        nMotion_Prev = 0
elif nPage == _PAGE_REMOTE_SPECIAL:
        Clear_All()

        btnList = [
            _BTN_MENU,
            _BTN_REMOTE_NORMAL,
            _BTN_REMOTE_FIGHT,
            _BTN_REMOTE_SPECIAL,
            #Rotate button
            _BTN_TURN_L,
            _BTN_TURN_R,
            #-Diagonal
            _BTN_MOVE_DG_FL,
            _BTN_MOVE_DG_FR,
```

Page 8

```
            _BTN_MOVE_DG_BL,
            _BTN_MOVE_DG_BR,
            #Go button
            _BTN_MOVE_UL,
            _BTN_MOVE_U,
            _BTN_MOVE_UR,
            _BTN_MOVE_L,
            _BTN_MOVE_R,
            _BTN_MOVE_DL,
            _BTN_MOVE_D,
            _BTN_MOVE_DR,
            _BTN_MOVE_DR,
            #Motion speed button
            _BTN_SPD,
            #Torque ON/OFF button
            _BTN_TORQ,
            #LED change button
            _BTN_LED,
            #Wake up button
            _BTN_GETUP,
            #Normal mode action button
            _BTN_ACT_01, #Hand greeting
            _BTN_ACT_02, #Bending greetings
            _BTN_ACT_03, # ceremony
            _BTN_ACT_04, # provocation
            _BTN_ACT_05, # mulgunamu
```

```
                _BTN_ACT_06, #Push-ups
                _BTN_ACT_07, # Applause
                _BTN_ACT_08 #Side roll
                ]
        Show_Background( 5 ) # + CVar.nSpeed)
        Motion_Ready( 1 )
        nMotion_Prev = 0
elif nPage == _PAGE_STREAM:
        Clear_All()

        btnList = [
                _BTN_MENU,
                #Rotate button
                _BTN_TURN_L,
                _BTN_TURN_R,
                #-Diagonal
                _BTN_MOVE_DG_FL,
                _BTN_MOVE_DG_FR,
                _BTN_MOVE_DG_BL,
                _BTN_MOVE_DG_BR,
                #Go button
                _BTN_MOVE_UL,
                _BTN_MOVE_U,
                _BTN_MOVE_UR,
                _BTN_MOVE_L,
```



```
                _BTN_MOVE_R,
                _BTN_MOVE_DL,
                _BTN_MOVE_D,
                _BTN_MOVE_DR,
                _BTN_MOVE_DR,
                #Motion speed button
                _BTN_SPD,
                #Torque ON/OFF button
                _BTN_TORQ,
                #LED change button
                _BTN_LED,
                #Wake up button
                _BTN_GETUP,
                #Normal mode action button
                _BTN_ACT_01, #Hand greeting
                _BTN_ACT_02, #Bending greetings
                _BTN_ACT_03, # ceremony
                _BTN_ACT_04, # provocation
                _BTN_ACT_05, # mulgunamu
                _BTN_ACT_06, #Push-ups
                _BTN_ACT_07, # Applause
                _BTN_ACT_08 #Side roll
                ]
```

```
        Show_Background( 11 + CVar.nSpeed)
        rpi.mode( 3 ) #0: Close, 1: Color, 2: Face Detection, 3: Streaming, 4: Marker, 5: Lane, 6: Emotion, 7: Hand
        smart.write8( 10700 , 1 ) # Smartphone streaming video screen 0: Close, 1: Display
        Motion_Ready( 2 )
        nMotion_Prev = 0
elif nPage == _PAGE_FACE_TRACK:
        Clear_All()

        btnList = [
                _BTN_MENU
                ]
        Show_Background( 13 )

        CVar.nPos_Servo = 512
        #Setup_Speed(254, 0)
        Motion_Ready( 2 )
        Motion_Play( 13 )
        #nMotion_Ready = 2 # Since the legs are in the open position, we need to set this to 2 so that the legs are removed in Motion_Ready().
          Collect motion

        OLLO( 1 , const.OLLO_JOINT_POSITION).write(CVar.nPos_Servo)

        rpi.mode( 2 ) #0: Close, 1: Color, 2: Face Detection, 3: Streaming, 4: Marker, 5: Lane, 6: Emotion, 7: Hand
        smart.write8( 10700 , 0 ) # Smartphone streaming video screen 0: Close, 1: Display

        # Timer initialization
        #tmrFace.Set()
```



```
        #CVar.IsTurning_Face = True # If you want to find the first face and say hello right away
          do. If false, the first face movement is required

elif nPage == _PAGE_MOTOR_TEST:
        Clear_All()
        btnList = [
                _BTN_MENU,
                _BTN_ID_01,
                _BTN_ID_02,
                _BTN_ID_03,
                _BTN_ID_04,
                _BTN_ID_05,
                _BTN_ID_06,
                _BTN_ID_07,
                _BTN_ID_08,
                _BTN_ID_09,
                _BTN_ID_10,
                _BTN_ID_11,
                _BTN_ID_12,
                _BTN_ID_13,
                _BTN_ID_14,
```

```
            _BTN_ID_15,
            _BTN_ID_16,
            _BTN_ID_17
            ]
        Show_Background( 14 )
        Show_Motors( -1 )

        # Initial position (knee upright position)
        #Setup_Speed(254, 0)
        #Motion_Ready(1)
        Motion_Play_And_Wait( 15 )
        Move_Center( True )
        #All LEDs Off
        DXL( 254 ).write8( 65 , 0 )

        Led( -1 ) # Off
elif nPage == _PAGE_OFFSET:
        Clear_All()

        btnList = [
            _BTN_MENU,
            _BTN_ID_01,
            _BTN_ID_02,
            _BTN_ID_03,
            _BTN_ID_04,
            _BTN_ID_05,
            _BTN_ID_06,
            _BTN_ID_07,
            _BTN_ID_08,
            _BTN_ID_09,
```



```
            _BTN_ID_10,
            _BTN_ID_11,
            _BTN_ID_12,
            _BTN_ID_13,
            _BTN_ID_14,
            _BTN_ID_15,
            _BTN_ID_16,
            _BTN_ID_17,
            _BTN_RESET,
            _BTN_INIT
            ]
        Show_Background( 14 )
        Show_Motors( -1 )
        # Reset & Init Pos
        Show_Image( 900 , 300 , 9 )
        # Initial position (knee upright position)
        #Setup_Speed(254, 0)
        #Motion_Ready(1)
        Motion_Play_And_Wait( 15 )
```

```
        # Center all motors
        Move_Center( True )
        Led( -1 ) # Off
elif nPage == _PAGE_OFFSET_DIALOG1:
        btnList = [
                _BTN_DIALOG_OK,
                _BTN_DIALOG_CANCEL,
                _BTN_DIALOG_TORQ,
                _BTN_DIALOG_PLUS,
                _BTN_DIALOG_MINUS
                ]
        Show_Dialog( 1 , CVar.nID)
elif nPage == _PAGE_OFFSET_DIALOG2:
        btnList = [
                _BTN_DIALOG_OK,
                _BTN_DIALOG_CANCEL
                ]
        Show_Dialog( 2 , CVar.nID)
elif nPage == _PAGE_OFFSET_DIALOG_CLOSE:
        btnList = [
                _BTN_MENU,
                _BTN_ID_01,
                _BTN_ID_02,
                _BTN_ID_03,
                _BTN_ID_04,
                _BTN_ID_05,
                _BTN_ID_06,
                _BTN_ID_07,
                _BTN_ID_08,
                _BTN_ID_09,
                _BTN_ID_10,
```



```
                _BTN_ID_11,
                _BTN_ID_12,
                _BTN_ID_13,
                _BTN_ID_14,
                _BTN_ID_15,
                _BTN_ID_16,
                _BTN_ID_17,
                _BTN_RESET,
                _BTN_INIT
                ]
        # Clear-dialog background
        Show_Dialog( 0 )
else :
        btnList = [
                _BTN_MENU
                ]
```

```
def Show_Motors(nSelect):
    #[off]11,12,13,... <-> [on]31,32,33,...
    for i in range( 0 , 17 ):
        nAdd = 0
        #Selected motor displays [On]
        if nSelect == i + 1 :
            nAdd = nAdd + 20
        Show_Image(round((btnList[i +1 ][ 0 ] + btnList[i +1 ][ 2 ]) /
          2 ),round((btnList[i +1 ][ 1 ] + btnList[i +1 ][ 3 ]) / 2 ), i + 11 + nAdd)

def GetUp():
    if (CVar.IsTorqOn == False ):
        TorqAll( True )
    nStandup = 0
    if (CVar.nPage != _PAGE_REMOTE_FIGHT) and (CVar.nPage != _PAGE_STREAM):
        nStandup = 100
    if (imu.pitch()> 6000 ): # fall forward
        Motion_Play_And_Wait( 3 )
        nStandup += 2
    elif (imu.pitch() < -6000 ): # fall backward
        Motion_Play_And_Wait( 4 )
        nStandup += 2

    if (nStandup> 100 ):
        Motion_Play_And_Wait( 16 )
    elif (nStandup == 100 ):
        Motion_Play_And_Wait( 1 )
    else :
        Motion_Play_And_Wait( 2 )

nMotion_Ready = -1
def Motion_Ready(nInit = 2 ):
    global nMotion_Ready
    # All motor torque On
    TorqAll( True )
```

Page 13

```
    # Update the entire motor position
    PositionUpdate()
    #Show_Num(400,500,nMotion_Ready)
    #Show_Num(600,500,nInit)

    if (nInit == 2 ):
        Motion_Play_And_Wait( 2 )
    else :
        if (nMotion_Ready < 0 ):
            Motion_Play_And_Wait( 2 )
            #Motion_Play_And_Wait(15) # Slow standing motion
            Motion_Play_And_Wait( 16 )
        else :
```

```
                if (nInit != nMotion_Ready):
                    if (nInit != 2 ):
                        Motion_Play_And_Wait( 16 )

            Motion_Play_And_Wait( 1 )
        nMotion_Ready = nInit

def Motion_Stop():
        Motion_Play( -3 )
nMotion_Prev = 0
def Motion_Play(nMotionIndex, nNextMotion = 0 ):
        global nMotion_Prev
        if (nMotionIndex <= 0 ):
            # Stop operation (0)
            #nMotionIndex => -3: End immediately, -2: End after executing step(Key-Frame),
              -1: Execute page (motion unit) and exit, 0: Execute page (motion unit) and exit
              End.
            motion.play((int)(nMotionIndex))
        else :
            Setup_Speed( 254 , 0 )
            if (nNextMotion == 0 ):
                motion.play((int)(nMotionIndex))
            else :
                motion.play((int)(nMotionIndex), (int)(nNextMotion))
        nMotion_Prev = nMotionIndex
def Motion_Play_And_Wait(nMotionIndex, nNextMotion = 0 ):
        Motion_Play(nMotionIndex, nNextMotion)
        Motion_Wait()
def Motion_Wait():
        while motion.status():
            if btnList != None :
                # Check button press-consumption of touch
                nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0, Event_Up1, Btn0,
                  Btn1 = GetButton(btnList)
def TorqAll(IsOn, IsSound = None ):
        TorqOnOff( -1 , IsOn, IsSound)
def TorqOnOff(nNum, IsOn, IsSound = None ):
```



```
        global nMotion_Ready
        if (IsOn == True ):
            if IsSound == True :
                buzzer.melody( 14 )
            if (nNum >= 0 ):
                DXL(nNum).torque_on()
            else :
                dxlbus.torque_on()
                CVar.IsTorqOn = True
        else :
            nMotion_Ready = -1
```

```
                CVar.IsTorqOn = False
                if IsSound == True :
                        buzzer.melody( 15 )
                if (nNum >= 0 ):
                        DXL(nNum).torque_off()
                else :
                        dxlbus.torque_off()

def GetResolution():
        for i in range( 0 , 100 ):
                screen smart.read32 = ( 10460 )
                CVar.nScreenWidth = screen & 0x0000FFFF
                CVar.nScreenHeight = (screen & 0xFFFF0000 ) >> 16
                if CVar.nScreenWidth> 0 and CVar.nScreenWidth < 65535 and
                  CVar.nScreenHeight> 0 and CVar.nScreenHeight < 65535 :
                        break

def GetTouch_Down():
        # 1 2 3 4 5
        # 6 7 8 9 10
        # 11 12 13 14 15
        # 16 17 18 19 20
        # 21 22 23 24 25
        if (smart.is_connected() == True ):
                XY_X0 = 0
                XY_Y0 = 0
                Pos_X0 = 0
                Pos_Y0 = 0
                Pos0 = 0
                XY_X1 = 0
                XY_Y1 = 0
                Pos_X1 = 0
                Pos_Y1 = 0
                Pos1 = 0

                # Touch-First
                Tmp = smart.read64( 10470 ) # Touch input coordinate
                nTouch0 = Tmp[ 0 ] & 0xffffffff
                nTouch1 = Tmp[ 1 ] & 0xffffffff
                IsChanged = False
```



```
                if (nTouch0> 0 ):
                        XY_X0 = nTouch0 & 0x0000FFFF
                        XY_Y0 = (nTouch0 >> 16 ) & 0x0000FFFF
                        Pos_X0 = (int)((XY_X0 / CVar.nScreenWidth) * 5 + 1 )
                        Pos_Y0 = (int)((XY_Y0 / CVar.nScreenHeight) * 5 + 1 )
                        Pos_X0 Pos0 + = (Pos_Y0 - 1 ) * 5
                        XY_X0 = (int)(XY_X0 * _RATIO / CVar.nScreenWidth)
                        XY_Y0 = (int)(XY_Y0 * _RATIO / CVar.nScreenHeight)
                        if (nTouch1> 0 ):
```

```
                    XY_X1 = nTouch1 & 0x0000FFFF
                    XY_Y1 = (nTouch1 >> 16 ) & 0x0000FFFF
                    Pos_X1 = (int)((XY_X1 / CVar.nScreenWidth) * 5 + 1 )
                    Pos_Y1 = (int)((XY_Y1 / CVar.nScreenHeight) * 5 + 1 )
                    Pos_X1 Pos1 + = (Pos_Y1 - 1 ) * 5
                    XY_X1 = (int)(XY_X1 * _RATIO / CVar.nScreenWidth)
                    XY_Y1 = (int)(XY_Y1 * _RATIO / CVar.nScreenHeight)

                CVar.nTouch_Pos0 = Pos0
                CVar.nTouch_Pos_X0 = Pos_X0
                CVar.nTouch_Pos_Y0 = Pos_Y0
                CVar.nTouch_XY_X0 = XY_X0
                CVar.nTouch_XY_Y0 = XY_Y0

                CVar.nTouch_Pos1 = Pos1
                CVar.nTouch_Pos_X1 = Pos_X1
                CVar.nTouch_Pos_Y1 = Pos_Y1
                CVar.nTouch_XY_X1 = XY_X1
                CVar.nTouch_XY_Y1 = XY_Y1
            else :
                CVar.nTouch_Pos0 = 0
                CVar.nTouch_Pos_X0 = 0
                CVar.nTouch_Pos_Y0 = 0
                CVar.nTouch_XY_X0 = 0
                CVar.nTouch_XY_Y0 = 0

                CVar.nTouch_Pos1 = 0
                CVar.nTouch_Pos_X1 = 0
                CVar.nTouch_Pos_Y1 = 0
                CVar.nTouch_XY_X1 = 0
                CVar.nTouch_XY_Y1 = 0
        else :
            IsPhone = False
# If you enter a coordinate value between 1 and 100, it is converted to a coordinate value suitable for a smartphone.
def Set(nX, nY):
        nResX = nX * CVar.nScreenWidth / _RATIO
        nResY = nY * CVar.nScreenHeight / _RATIO
        return nResX, nResY

def IsButton(nX, nY, Btn):
        if (Btn[ 0 ] < 0 ):
            right = (Btn[ 0 ] * -200 )
```



```
            left = right- 200
            bottom = (Btn[ 1 ] * -200 )
            top = bottom- 200
            if ((nY >= top) and (nY <= bottom)):
                if ((nX >= left) and (nX <= right)):
                    return True
```

```
        else :  if ((nY >= Btn[ 1 ]) and (nY <= Btn[ 3 ])):
                    if ((nX >= Btn[ 0 ]) and (nX <= Btn[ 2 ])):
                        return True
        return False

# Update motor position
def PositionUpdate():
        etc.write8( 65 , 3 )
        while ( True ):
            if etc.read8( 65 ) == 0 :
                break

#Print number simple test
def Show_Num(nX, nY, nValue, nColor = _COLOR_NONE, nSize = None ):
        if (nSize == 0 ):
            Clear_Num(nX, nY)
        elif (nColor == _COLOR_NONE):
            Show(_SHOW_NUM, nX, nY, 60 , _COLOR_RED, nValue)
        elif (nSize == None ):
            Show(_SHOW_NUM, nX, nY, 60 , nColor, nValue)
        else :
            Show(_SHOW_NUM, nX, nY, nSize, nColor, nValue)
def Show_Point(nX, nY, nColor, nSize = None ):
        if (nSize == None ):
            Show(_SHOW_SHAPE, nX, nY, 20 , nColor, 1 )
        else :
            Show(_SHOW_SHAPE, nX, nY, nSize, nColor, 1 )
def Show_Text(nX, nY, nValue, nColor = _COLOR_NONE, nSize = None ):
        if (nColor == _COLOR_NONE):
            Show(_SHOW_TEXT, nX, nY, 60 , _COLOR_RED, nValue)
        elif (nSize == None ):
            Show(_SHOW_TEXT, nX, nY, 60 , nColor, nValue)
        else :
            Show(_SHOW_TEXT, nX, nY, nSize, nColor, nValue)
def Show_Image(nX, nY, nValue, nSize = None ):
        if (nSize == None ):
            Show(_SHOW_IMAGE, nX, nY, 1 , 0 , nValue)
        else :
            Show(_SHOW_SHAPE, nX, nY, nSize, 0 , nValue)
def Show_Background(nValue):
        if (nValue != CVar.nBack_Background):
            smart.display.back_image(nValue)
            CVar.nBack_Background = nValue
#nShowType: 0-picture, 1-letter, 2-shape, 3-number
```



```
# nValue ([Image-Index], [Shape-1: Circle, 2: Square, 3: Triangle], [Text-Index], [Num-
  Value])
# nColor (0: Unknown, 1: White, 2: Black, 3: Red, 4: Green, 5: Blue 6: Yellow, 7: Light Gray,
  8: gray, 9: dark gray)
```

```
def Show(nShowType, nX, nY, nSize, nColor, nValue):
    if ((nShowType >= 0 ) and (nShowType < 4 )):
        nX, nY = Set(nX, nY)
        smart.write32( 10480 , int(nX) | (int(nY) << 16 ))
        nTmp = nValue * 256 + nSize * 65536 + nColor * 16777216
        if (nShowType == _SHOW_IMAGE): # Do not use the color value of 0xff000000 digits
            smart.display.front_image(nTmp & 16777215 ) # & 0xffffff(=16777215)
        elif (nShowType == _SHOW_SHAPE):
            smart.display.shape(nTmp)
        elif (nShowType == _SHOW_TEXT):
            smart.display.text(nTmp)
        elif (nShowType == _SHOW_NUM):
            smart.display.number(nTmp)
def Clear_All():
    smart.write32( 10480 , 0 )
    smart.display.front_image( 0 )
    smart.display.shape( 0 )
    smart.display.text( 0 )
    smart.display.number( 0 )
def Clear_Shape():
    smart.write32( 10480 , 0 )
    smart.display.shape( 0 )
def Clear_Text():
    smart.write32( 10480 , 0 )
    smart.display.text( 0 )
def Clear_Num(nX = None , nY = None ):
    if (nX == None ) or (nY == None ):
        smart.write32( 10480 , 0 )
        smart.display.number( 0 )
    else :
        Show(_SHOW_NUM, nX, nY, 0 , 0 , 0 )

# [nRed: 0~100], [nBlue: 0~100]
def LedPwm(nRed, nBlue):
    OLLO( 2 , const.OLLO_RED_BRIGHTNESS).write(round(nRed)) # 0~100
    OLLO( 2 , const.OLLO_BLUE_BRIGHTNESS).write(round(nBlue)) # 0~100

def Led(nValue = 0 ):
    red_led = OLLO( 2 , const.OLLO_RED_BRIGHTNESS)
    blue_led = OLLO( 2 , const.OLLO_BLUE_BRIGHTNESS)
    nRed = 0
    nBlue = 0
    if nValue == 0 :
        if red_led.read()> 0 and blue_led.read()> 0 :
            pass
        elif red_led.read() <= 0 and blue_led.read() <= 0 :
            nRed = 100
```



```
        elif red_led.read()> 0 and blue_led.read() <= 0 :
            nBlue = 100
```

```
                else :
                        nRed = 100
                        nBlue = 100
                red_led.write(nRed) # 0~100
                blue_led.write(nBlue) # 0~100
        elif nValue < 0 :
                red_led.write( 0 ) # 0~100
                blue_led.write( 0 ) # 0~100
        # manual change
        else :
                #nRed = 100 * (nValue% 2)
                #nBlue = int(nValue / 2)
                red_led.write( 100 * (nValue% 2 )) # 0~100
                blue_led.write(int(nValue / 2 )) # 0~100

# Function that converts angle to data used in motor (float -> int)
def CalcAngle2Raw(fAngle):
        if fAngle == None :
                fAngle = 0.0
        return (int)(round(fAngle * 4096.0 / 360.0 + 2048.0 ))
# Function that converts the data used in the motor into an angle value (int -> float)
def CalcRaw2Angle(nRaw):
        if nRaw == None :
                nRaw = 0
        return (float)( 360.0 * ((nRaw- 2048.0 ) / 4096.0 ))
def Offset_GetData(nID):
        nValue = etc.read16( 200 + nID * 2 )
        if (nValue> 32767 ):
                nValue = ( 65536 -nValue) * -1
        return int(nValue)
def Offset_Read():
        global aOffset
        etc.write8( 199 , 1 )
        delay( 50 )
        nMot_First = 17
        nMot_Cnt = 1
        #print("Offset=")
        for i in range(nMot_First,nMot_First + nMot_Cnt):
                nID = i #i + 1
                aOffset[nID] = Offset_GetData(nID)
                #print(nID, aOffset[nID])
def Offset_Write():
        etc.write8( 199 , 2 )
def Offset_Clear():
        global aOffset
        etc.write8( 199 , 3 )
        #aOffset.clear()
        nCnt = len(aOffset)
        aOffset = [ 0 ] * nCnt
```

```
    #aOffset = [0 for i in range(nCnt)]
def Test1(nBackgroundImage = 1 , nImage = 0 , x= 500 ,y= 500 ):
    pass

'''
    # Print background image
    if (CVar.nTouch_Pos0> 0):
        # At the moment of touch, all basic shapes are erased.
        Clear_All()

    # Basic figures and drawings to hear ...
    Show_Background(nBackgroundImage)
    #Test Image: Show_Image(x, y, image index)
    if (x>0) and (y>0) and (nImage>0):
        Show_Image(x, y, nImage)

    if (CVar.nTouch_Pos0> 0):
        Clear_Num()
        Clear_Shape()
        # X
        nPos = 50
        nGap = 20

        nX = CVar.nTouch_XY_X0
        nY = CVar.nTouch_XY_Y0
        if nX <100:
            nX = 100
        elif nX> 900:
            nX = 900
        if nY <100:
            nY = 100
        elif nY> 900:
            nY = 900

        Show_Num(nX-nPos-nGap, nY-40, (int)(CVar.nTouch_XY_X0 / 100%
          10))
        Show_Num(nX-nPos,                                   nY-40, (int)(CVar.nTouch_XY_X0 / 10% 10))
        Show_Num(nX-nPos + nGap, nY-40, (int)(CVar.nTouch_XY_X0% 10))
        # Y
        nPos = 50
        Show_Num(nX + nPos-nGap, nY-40, (int)(CVar.nTouch_XY_Y0 / 100%
          10))
        Show_Num(nX + nPos,                                 nY-40, (int)(CVar.nTouch_XY_Y0 / 10% 10))
        Show_Num(nX + nPos + nGap, nY-40, (int)(CVar.nTouch_XY_Y0% 10))

        Show_Num(nX-30, nY+40, CVar.nTouch_Pos_X0, _COLOR_GREEN)
        Show_Num(nX+30, nY+40, CVar.nTouch_Pos_Y0, _COLOR_GREEN)

        Show_Point(CVar.nTouch_XY_X0, CVar.nTouch_XY_Y0, _COLOR_BLUE)

    if (CVar.nTouch_Pos1> 0):
```

```
                # X
                nPos = 50
                nGap = 20

                nX = CVar.nTouch_XY_X1
                nY = CVar.nTouch_XY_Y1
                if nX <100:
                        nX = 100
                elif nX> 900:
                        nX = 900
                if nY <100:
                        nY = 100
                elif nY> 900:
                        nY = 900

                Show_Num(nX-nPos-nGap, nY-40, (int)(CVar.nTouch_XY_X1 / 100%
                  10))
                Show_Num(nX-nPos,                                nY-40, (int)(CVar.nTouch_XY_X1 / 10% 10))
                Show_Num(nX-nPos + nGap, nY-40, (int)(CVar.nTouch_XY_X1% 10))
                # Y
                nPos = 50
                Show_Num(nX + nPos-nGap, nY-40, (int)(CVar.nTouch_XY_Y1 / 100%
                  10))
                Show_Num(nX + nPos,                              nY-40, (int)(CVar.nTouch_XY_Y1 / 10% 10))
                Show_Num(nX + nPos + nGap, nY-40, (int)(CVar.nTouch_XY_Y1% 10))

                Show_Num(nX-30, nY+40, CVar.nTouch_Pos_X1, _COLOR_GREEN)
                Show_Num(nX+30, nY+40, CVar.nTouch_Pos_Y1, _COLOR_GREEN)

                Show_Point(CVar.nTouch_XY_X1, CVar.nTouch_XY_Y1, _COLOR_RED)
'''
#Set the motor operation speed (Based on Position): 0 initialization
def Setup_Speed(nID, nValue):
        DXL(nID).write32( 112 , nValue) # 112: profile velocity
# Align all motors to the center (Motion_Offset)
def Move_Center(IsOffset = False ):
        #profile speed adjustment
        Setup_Speed( 254 , 20 )
        #syncwrite
        etc.write8( 1200 , 0 )
        etc.write16( 1202 , 116 )
        etc.write8( 1204 , 4 )
        nMot_First = 1
        nMot_Cnt = 17
        for i in range(nMot_First,nMot_First + nMot_Cnt):
                nID = i #i + 1
                nValue = 0
                if (IsOffset):
                        nValue = Offset_GetData(nID)
                etc.write8( 1205 ,nID)
```

```
                etc.write32( 1206 , 2048 + nValue)
                etc.write8( 1200 , 1 )

        etc.write8( 1200 , 2 )
        #profile speed restore
        Setup_Speed( 254 , 0 )
        # Align all motors to the center (Motion_Offset)

def TestMove(nID, fAngle, nSpeed = -1 , IsOffset = False ):
        if (nSpeed >= 0 ):
                Setup_Speed(nID, nSpeed)
        DXL(nID).write32( 116 , CalcAngle2Raw(fAngle) + aOffset[nID]) # 116: Goal
          position

        tmr = CTimer()
        tmr.Set()
        nPass = 0
        while ( True ):
                nMoving = DXL(nID).read8( 122 )
                if ((tmr.Get() >= 200 ) and (nPass == 0 )):
                        break
                elif (nMoving != 0 ):
                        nPass = 1
                elif ((nMoving == 0 ) and (nPass == 1 )):
                        nPass = 2
                        break
        #tmr = None
def WaitButtonUp(nTouchMode = 0 ): # 0: All, 1: FirstTouch, 2: SecondTouch
        while ( True ):
                # Check button press
                nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0, Event_Up1, Btn0, Btn1 =
                  GetButton(btnList)
                if (nTouchMode == 1 ):
                        if (nNum0 < 0 ):
                                break
                if (nTouchMode == 2 ):
                        if (nNum1 < 0 ):
                                break
                else :
                        if ((nNum0 < 0 ) and (nNum1 < 0 )):
                                break

def GetButton(btns):
        #Check the touch input of the smartphone
        GetTouch_Down()

        nNum0 = -1
        nNum1 = -1
```

```
nDown0 = 0
nDown1 = 0
nUp0 = 0
```



```
nUp1 = 0
Btn0 = None
Btn1 = None
nNum = 0
nCnt = 0
if (CVar.nTouch_Pos0> 0 ):
        nCnt = nCnt + 1
if (CVar.nTouch_Pos1> 0 ):
        nCnt = nCnt + 1

nPass = 0
for btn in btns:
        if (CVar.nTouch_Pos0> 0 ):
                if (IsButton(CVar.nTouch_XY_X0, CVar.nTouch_XY_Y0, btn) == True ):
                        nNum0 = btn[_BTN_INDEX]
                        Btn0 = btn
                        nPass = nPass | 0x01
        if (CVar.nTouch_Pos1> 0 ):
                if (IsButton(CVar.nTouch_XY_X1, CVar.nTouch_XY_Y1, btn) == True ):
                        nNum1 = btn[_BTN_INDEX]
                        Btn1 = btn
                        nPass = nPass | 0x10
        if (nCnt == 1 ):
                if (nPass> 0 ):
                        break
        else :
                if (nPass == 0x11 ):
                        break
        nNum = nNum + 1

if ((nNum1 == nNum0) and (nNum0 >= 0 )):
        nNum1 = -1

# switching
if (((nNum0 >= 0 ) and (nNum0 == CVar.nButton_1)) or ((nNum1 >= 0 ) and
  (nNum1 == CVar.nButton_0))):
        nNum2 = nNum1
        nNum1 = nNum0
        nNum0 = nNum2
#Button Down Event
if (nNum0 >= 0 ):
        if (CVar.nButton_0 != nNum0):
                nDown0 = 1
        CVar.btn0 = Btn0
else :
        if (CVar.nButton_0 >= 0 ):
```

```
                nUp0 = 1
        if (nNum1 >= 0 ):
                if (CVar.nButton_1 != nNum1):
                        nDown1 = 1
                CVar.btn1 = Btn1
```



```
        else :
                if (CVar.nButton_1 >= 0 ):
                        nUp1 = 1
        CVar.nButton_0 = nNum0
        CVar.nButton_1 = nNum1
        if nUp0 == 1 :
                Btn0 = CVar.btn0
        if nUp1 == 1 :
                Btn1 = CVar.btn1
        return nNum0, nNum1, nDown0, nDown1, nUp0, nUp1, Btn0, Btn1
def CheckButton(Btn0, Btn1, Btn):
        if (Btn0 == Btn):
                return 1
        elif (Btn1 == Btn):
                return 2
        return 0
def CheckButton_Event(Btn0, Btn1, nEvent0, nEvent1, Btn):
        if (Btn0 == Btn):
                if (nEvent0> 0 ):
                        return 1
        elif (Btn1 == Btn):
                if (nEvent1> 0 ):
                        return 2
        return 0
def Page_MenuBar(nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0, Event_Up1,
  Btn0, Btn1):
        if (Btn0 == _BTN_SUB_OUT):
                ShowPage(CVar.nPage_Prev)
        #When pressing the outside part while the menu is displayed
        elif (Btn0 == _BTN_SUB_REMOTE):
                ShowPage(_PAGE_REMOTE_NORMAL)
        elif (Btn0 == _BTN_SUB_STREAM):
                ShowPage(_PAGE_STREAM)
        elif (Btn0 == _BTN_SUB_FACE_TRACK):
                ShowPage(_PAGE_FACE_TRACK)
        elif (Btn0 == _BTN_SUB_MOTOR):
                ShowPage(_PAGE_MOTOR_TEST)
        elif (Btn0 == _BTN_SUB_OFFSET):
                ShowPage(_PAGE_OFFSET)
        elif (Btn0 == _BTN_SUB_X):
                ShowPage(CVar.nPage_Prev)
        #Check the touch input of the smartphone
        if ((nNum0 >= 0 ) or (nNum1 >= 0 )):
                WaitButtonUp()
```

```
#Knee ID = 15, 16
def CheckKnee():
        CVar.nKnee = 0
        fAngle = CalcRaw2Angle(DXL( 15 ).present_position())
        if (fAngle> 100.0 ):
                CVar.nKnee = CVar.nKnee + 1
```

Page 24

```
        # Since the assembly direction is opposite, the same direction can be analyzed if the sign is (-) when converting the angle (inverted).

        fAngle = -CalcRaw2Angle(DXL( 16 ).present_position())
        if (fAngle> 100.0 ):
                CVar.nKnee = CVar.nKnee + 1
def CheckTouch_for_walking(nDirection):
        CVar.nWalking = 0
        if ((nDirection >= 12 ) and (nDirection <= 14 )):
                # 12 (forward-left) 13 (forward) 14 (forward-right)
                CVar.nWalking = 1
        elif ((nDirection >= 17 ) and (nDirection <= 19 )):
                # 17 (Backward-left) 18 (Backward) 19 (Backward-right)
                CVar.nWalking = -1
def walking(nDirection):
        CVar.nWalking = 0
        if (nDirection> 0 ):
                CheckTouch_for_walking(nDirection)
                nMotion = 0
                nMotion_Next = 0

                _MOTION_KNEE = 201
                _MOTION = 101
                if (CVar.nPage == _PAGE_REMOTE_SPECIAL):
                        _MOTION_KNEE = 301
                        _MOTION = 301
                        CVar.nSpeed = 0

                if (CVar.nWalking != 0 ):
                        #if (CVar.nWalking != CVar.nWalking_Prev):
                        # Vibration()
                        if (CVar.nKnee> 0 ):
                                _MOTION_KNEE nMotion = - (CVar.nWalking - 1 ) * 5 / 2
                                _MOTION_KNEE nMotion_Next = - (CVar.nWalking - 1 ) * 5 / 2 +
                                  (nDirection- 11 )% 5
                        else :
                                nMotion = _MOTION + CVar.nSpeed * 50- (CVar.nWalking- 1 ) * 5
                                  Of 2
                                nMotion_Next = _MOTION + CVar.nSpeed * 50- (CVar.nWalking-
                                  1 ) * 5 / 2 + (nDirection - 11 )% 5
                        Motion_Play(nMotion, nMotion_Next)
```

```
nCnt = 0
CVar.nWalking_Prev = CVar.nWalking
while (CVar.nWalking != 0 ):
        IsPunch = False
        while (motion.status() == True ):
                # Control mode
                nDirection = 0
                if (btnList != None ):
                        nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0,
                          Event_Up1, Btn0, Btn1 = GetButton(btnList)
```



```
                        if CheckButton(Btn0, Btn1, _BTN_MOVE_UL):
                                nDirection = 12
                        if CheckButton(Btn0, Btn1, _BTN_MOVE_U):
                                nDirection = 13
                        if CheckButton(Btn0, Btn1, _BTN_MOVE_UR):
                                nDirection = 14
                        if CheckButton(Btn0, Btn1, _BTN_MOVE_DL):
                                nDirection = 17
                        if CheckButton(Btn0, Btn1, _BTN_MOVE_D):
                                nDirection = 18
                        if CheckButton(Btn0, Btn1, _BTN_MOVE_DR):
                                nDirection = 19
                        if CheckButton(Btn0, Btn1, _BTN_ACT_01):
                                IsPunch = True
                                #print("Punch")
                CheckTouch_for_walking(nDirection)
        if (CVar.nWalking != 0 ):
                if (CVar.nWalking != CVar.nWalking_Prev):
                        CVar.nWalking = CVar.nWalking_Prev
                        break
                nCnt = 1
        else :
                nCnt = 0
        if (nCnt> 0 ):
                nCnt = nCnt- 1
        else :
                break
        if (IsPunch):
                break
        else :
                # ready: 100, start: 101, left: 102, go: 103, right: 104,
                  end: 105
                if (CVar.nKnee> 0 ):
                        _MOTION_KNEE nMotion = - (CVar.nWalking - 1 ) * 5 / 2 +
                          (nDirection- 11 )% 5
                        nMotion_Next = nMotion
                else :
                        nMotion = _MOTION + CVar.nSpeed * 50- (CVar.nWalking
                          - 1 ) * 5 / 2 + (nDirection - 11 )% 5
```

```
                                nMotion_Next = nMotion
                            Motion_Play(nMotion, nMotion_Next)
                            CVar.nWalking_Prev = CVar.nWalking
                    if (CVar.nWalking_Prev != CVar.nWalking):
                        CVar.nWalking = CVar.nWalking_Prev
                    if (IsPunch):
                        nMotion = 80
                    else :
                        if (CVar.nKnee> 0 ):
                            NMotion = (_MOTION_KNEE + 4 ) - (CVar.nWalking - 1 ) * 5 / 2
                        else :
```



```
                            NMotion = (_MOTION + 4 ) + CVar.nSpeed * 50 -
                              (CVar.nWalking - 1 ) * 5 / 2
                    #Motion_Play(nMotion)
                    #Motion_Wait()
                    Motion_Play_And_Wait(nMotion)
                    if (CVar.nPage != _PAGE_REMOTE_FIGHT) and (CVar.nPage !=
                      _PAGE_STREAM):
                        Motion_Play_And_Wait( 16 )
def Filter_LowPass(fWeight, fVal_Curr, fVal_Prev):
    return int(round(float(fVal_Curr) * fWeight + ( 1.0 -fWeight) *
      float(fVal_Prev), 0 ))

def Page_Remote(nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0, Event_Up1,
  Btn0, Btn1):
    global nMotion_Prev
    # Switch each page in remote control mode
    if Event_Dn0 == 1 :
        IsShowPage = True
        #print("Btn0={0},{1}".format(nNum0, nNum1))
        if (Btn0 == _BTN_REMOTE_NORMAL):
            ShowPage(_PAGE_REMOTE_NORMAL)
        elif (Btn0 == _BTN_REMOTE_FIGHT):
            ShowPage(_PAGE_REMOTE_FIGHT)
        elif (Btn0 == _BTN_REMOTE_SPECIAL):
            ShowPage(_PAGE_REMOTE_SPECIAL)
        else :
            IsShowPage = False
        if IsShowPage == True :
            #Because the button event is reset as the page changes, the event can occur again.
            #Before the page changes, you need to write a waiting statement that waits for the button to be released.
            WaitButtonUp( 1 )                    # Wait for the button to be released

    #Torque ON/OFF button-Since torque off is a function to be careful, when only one touch comes in
      Let it work.
    if (Btn0 == _BTN_TORQ): # or (Btn1 == _BTN_TORQ):
        if Event_Dn0 == 1 :
```

```
                tmrTorqBtn.Set()
                if (CVar.IsTorqOn == True ):
                        #Sit down before you release the talk.
                        Motion_Play_And_Wait( 60 )
                        delay( 100 )
                        TorqAll( False , True )

                        nMotion_Prev = 0
                else :
                        TorqAll( True , True )
                        if (CVar.nPage == _PAGE_REMOTE_FIGHT) or (CVar.nPage ==
                          _PAGE_STREAM):
                                Motion_Ready( 2 )
                        else :
                                Motion_Ready( 1 )
```



```
        elif Event_Up0 == 1 :
                tmrTorqBtn.Destroy()
        ##If you keep holding down
        else :
                #Reboot All
                if (tmrTorqBtn.Get() >= 3000 ):
                        #Command-Reboot
                        dxlbus.reboot()
                        #Reboot-Beep
                        buzzer.melody( 1 )
                        #TorqOff variable
                        CVar.IsTorqOn = False
                        # No more timer detection.
                        tmrTorqBtn.Destroy()

if CheckButton_Event(Btn0, Btn1, Event_Dn0, Event_Dn1, _BTN_LED):
        Led()
if CheckButton(Btn0, Btn1, _BTN_GETUP):
        GetUp()
Speed adjustment is not performed in #Special Page.
if (CVar.nPage != _PAGE_REMOTE_SPECIAL):
        if ((Btn0 == _BTN_SPD) and (Event_Dn0 == 1 )) or ((Btn1 == _BTN_SPD)
          and (Event_Dn1 == 1 )):
                CVar.nSpeed = (CVar.nSpeed + 1 )% 2
                Show_Background((CVar.nPage* 2 ) + CVar.nSpeed- 1 )
#else:
# CVar.nSpeed = 0
################################## ####
# Control mode
nDirection = 0
if CheckButton(Btn0, Btn1, _BTN_MOVE_UL):
        nDirection = 12
if CheckButton(Btn0, Btn1, _BTN_MOVE_U):
        nDirection = 13
```

```
if CheckButton(Btn0, Btn1, _BTN_MOVE_UR):
        nDirection = 14
if CheckButton(Btn0, Btn1, _BTN_MOVE_DL):
        nDirection = 17
if CheckButton(Btn0, Btn1, _BTN_MOVE_D):
        nDirection = 18
if CheckButton(Btn0, Btn1, _BTN_MOVE_DR):
        nDirection = 19
walking(nDirection)
################################## ####
if nDirection == 0 :
        nAction = 0
        # Turn left/right
        if CheckButton(Btn0, Btn1, _BTN_TURN_L):
                if (CVar.nPage == _PAGE_REMOTE_SPECIAL):
                        nAction = 90 #350
                else :
```



```
                        nAction = 50 + CVar.nSpeed * 2                  # 50, 52
        elif CheckButton(Btn0, Btn1, _BTN_TURN_R):
                if (CVar.nPage == _PAGE_REMOTE_SPECIAL):
                        nAction = 91 #351
                else :
                        nAction = 51 + CVar.nSpeed * 2                  # 51, 53
        ################################## ####
        # Diagonal movement
        # ↖
        elif CheckButton(Btn0, Btn1, _BTN_MOVE_DG_FL):
                if (CVar.nPage == _PAGE_REMOTE_SPECIAL):
                        nAction = 94
                else :
                        nAction = 70
        # ↗
        elif CheckButton(Btn0, Btn1, _BTN_MOVE_DG_FR):
                if (CVar.nPage == _PAGE_REMOTE_SPECIAL):
                        nAction = 95
                else :
                        nAction = 71
        # ↙
        elif CheckButton(Btn0, Btn1, _BTN_MOVE_DG_BL):
                if (CVar.nPage == _PAGE_REMOTE_SPECIAL):
                        nAction = 96
                else :
                        nAction = 72
        # ↘
        elif CheckButton(Btn0, Btn1, _BTN_MOVE_DG_BR):
                if (CVar.nPage == _PAGE_REMOTE_SPECIAL):
                        nAction = 97
                else :
```

```
                nAction = 73
################################## ####
# Sidewalk
elif CheckButton(Btn0, Btn1, _BTN_MOVE_L):
        if (CVar.nPage == _PAGE_REMOTE_SPECIAL):
                nAction = 92 #354
        else :
                nAction = 54 + CVar.nSpeed * 2                  # left
elif CheckButton(Btn0, Btn1, _BTN_MOVE_R):
        if (CVar.nPage == _PAGE_REMOTE_SPECIAL):
                nAction = 93 #355
        else :
                nAction = 55 + CVar.nSpeed * 2                  # right
################################## ####
elif (CVar.nPage == _PAGE_REMOTE_NORMAL):
        if CheckButton(Btn0, Btn1, _BTN_ACT_01):
                nAction = 5
        elif CheckButton(Btn0, Btn1, _BTN_ACT_02):
                nAction = 6
        elif CheckButton(Btn0, Btn1, _BTN_ACT_03):
```



```
                nAction = 7
        elif CheckButton(Btn0, Btn1, _BTN_ACT_04):
                nAction = 8
        elif CheckButton(Btn0, Btn1, _BTN_ACT_05):
                nAction = 10
        elif CheckButton(Btn0, Btn1, _BTN_ACT_06):
                nAction = 12
        elif CheckButton(Btn0, Btn1, _BTN_ACT_07):
                nAction = 9
        elif CheckButton(Btn0, Btn1, _BTN_ACT_08):
                nAction = 11
elif (CVar.nPage == _PAGE_REMOTE_FIGHT):
        if CheckButton(Btn0, Btn1, _BTN_ACT_01):
                nAction = 21
                Move_Offset1( 17 , smart.sensor.tilt_right()-
                  smart.sensor.tilt_left())
                Move_Offset1( 3 , smart.sensor.tilt_down()-
                  smart.sensor.tilt_up())
                #Move_Offset1(4, smart.sensor.tilt_right()-
                  smart.sensor.tilt_left())
        elif CheckButton(Btn0, Btn1, _BTN_ACT_02):
                nAction = 22
                Move_Offset1( 17 , smart.sensor.tilt_right()-
                  smart.sensor.tilt_left())
                Move_Offset1( 1 , smart.sensor.tilt_up()-
                  smart.sensor.tilt_down())
                # Move_Offset1(2, smart.sensor.tilt_left()-
                  smart.sensor.tilt_right())
        elif CheckButton(Btn0, Btn1, _BTN_ACT_03):
```

```
                nAction = 23
                #Move_Offset1(3, smart.sensor.tilt_down()-
                  smart.sensor.tilt_up())
                #Move_Offset1(4, smart.sensor.tilt_right()-
                  smart.sensor.tilt_left())
            elif CheckButton(Btn0, Btn1, _BTN_ACT_04):
                nAction = 24
                #Move_Offset1(1, smart.sensor.tilt_up()-
                  smart.sensor.tilt_down())
                #Move_Offset1(2, smart.sensor.tilt_left()-
                  smart.sensor.tilt_right())
            elif CheckButton(Btn0, Btn1, _BTN_ACT_05):
                nAction = 25
                ##Move_Offset1(3, smart.sensor.tilt_down()-
                  smart.sensor.tilt_up())
                ##Move_Offset1(4, smart.sensor.tilt_right()-
                  smart.sensor.tilt_left())
            if CheckButton(Btn0, Btn1, _BTN_ACT_06):
                nAction = 26

                ##Move_Offset1(1, smart.sensor.tilt_up()-
                  smart.sensor.tilt_down())
```



```
                ##Move_Offset1(2, smart.sensor.tilt_left()-
                  smart.sensor.tilt_right())

            if CheckButton(Btn0, Btn1, _BTN_ACT_07):
                nAction = 27
                #Move_Offset1(3, smart.sensor.tilt_down()-
                  smart.sensor.tilt_up())
                #Move_Offset1(4, smart.sensor.tilt_right()-
                  smart.sensor.tilt_left())
            if CheckButton(Btn0, Btn1, _BTN_ACT_08):
                nAction = 28
                #Move_Offset1(1, smart.sensor.tilt_up()-
                  smart.sensor.tilt_down())
                #Move_Offset1(2, smart.sensor.tilt_left()-
                  smart.sensor.tilt_right())
            if CheckButton(Btn0, Btn1, _BTN_ACT_09):
                #nAction = 20
                WaitButtonUp()
                Defense( 20 )
            if CheckButton(Btn0, Btn1, _BTN_ACT_10):
                nAction = 29
        elif (CVar.nPage == _PAGE_STREAM):
            if CheckButton(Btn0, Btn1, _BTN_ACT_01):
                nAction = 21
            elif CheckButton(Btn0, Btn1, _BTN_ACT_02):
                nAction = 22
            elif CheckButton(Btn0, Btn1, _BTN_ACT_03):
```

```
                nAction = 23
        elif CheckButton(Btn0, Btn1, _BTN_ACT_04):
                nAction = 24
        elif CheckButton(Btn0, Btn1, _BTN_ACT_05):
                nAction = 25
        if CheckButton(Btn0, Btn1, _BTN_ACT_06):
                nAction = 26
        if CheckButton(Btn0, Btn1, _BTN_ACT_07):
                nAction = 27
        if CheckButton(Btn0, Btn1, _BTN_ACT_08):
                nAction = 28
elif (CVar.nPage == _PAGE_REMOTE_SPECIAL):
        if CheckButton(Btn0, Btn1, _BTN_ACT_01):
                pass
        if CheckButton(Btn0, Btn1, _BTN_ACT_02):
                pass
        if CheckButton(Btn0, Btn1, _BTN_ACT_03):
                pass
        if CheckButton(Btn0, Btn1, _BTN_ACT_04):
                pass
if (nAction> 0 ):
        #if (CVar.nPage != _PAGE_REMOTE_FIGHT) and (CVar.nPage !=
          _PAGE_STREAM):
        # if (nMotion_Prev == 0):
```

Page 31

```
        #                       if ((nAction >= 50) and (nAction <300)):
        #                               Motion_Play_And_Wait(2)
        if (CVar.nPage == _PAGE_REMOTE_NORMAL):
                if (nMotion_Prev == 0 ):
                        if (nAction >= 50 ):
                                Motion_Play_And_Wait( 2 )

        # #fLeft = smart.sensor.tilt_left() * 2
        # #fRight = smart.sensor.tilt_right() * 2
        # #fH = float(fUp-fDn) / 100.0
        # #fW = float(fRight-fLeft) / 100.0
        # #print(fW)
        # Move_Offset1(2, smart.sensor.tilt_left(),
          smart.sensor.tilt_right())
        # #print(smart.sensor.tilt_left(), smart.sensor.tilt_right())


        if (CVar.nPage == _PAGE_REMOTE_NORMAL):
                LedPwm( 50 , 100 )

        Motion_Play_And_Wait(nAction)
        #for i in range(1,17):
        #                       Move_Offset1(i, 0)
        #if (nAction == 23) or (nAction == 24):
```

```
                    # Move_Offset1(2, 0)
                    #if (nAction == 23) or (nAction == 24):
                    if (CVar.nPage == _PAGE_REMOTE_FIGHT):
                        for i in range( 1 , 18 ):
                            Move_Offset1(i, 0 )

                    if (CVar.nPage == _PAGE_REMOTE_NORMAL):
                        LedPwm( 0 , 0 )

                else :
                    if (CVar.nPage == _PAGE_REMOTE_NORMAL):
                        if (nMotion_Prev >= 50 ):
                            Motion_Play_And_Wait( 16 )
                    nMotion_Prev = 0
#def Deg2Rad(Angle):
# return Angle * math.pi / 180.0
#def Sin(fAngle):
# return math.sin(Deg2Rad(fAngle))
m_IsFind = False
def Page_Face_Track(nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0, Event_Up1,
  Btn0, Btn1):
    global m_IsFind
    CVar.m_nPwm = (CVar.m_nPwm + 10 )% 200
    #Value = int(100 * Sin(CVar.m_nPwm/200 * 180))
    Value = CVar.m_nPwm-CVar.m_nPwm% 100 * int(CVar.m_nPwm / 100 ) * 2
    if (rpi.area()> 0 ):
        Show_Text( 500 , 500 , 1 , _COLOR_BLUE, 140 )
```

Page 32

```
                #The value that the robot looks to the left converges to 0
                Rpi.position_x nPos_X = () - 320 / 2 # * 10/320
                #print(nPos_X)
                #Below y converges to 0
                #nPos_Y = rpi.position_y()# * 10/240
                nCenterGap = 0

                #nVal_Prev = CVar.nPos_Servo

                if (nPos_X <(- 20 + nCenterGap)):
                    #CVar.nPos_Servo = CVar.nPos_Servo + 0.3
                    CVar.nPos_Servo += 1
                    CVar.IsTurning_Face = True
                elif (nPos_X> ( 20 + nCenterGap)):
                    #CVar.nPos_Servo = CVar.nPos_Servo-0.3
                    CVar.nPos_Servo -= 1
                    CVar.IsTurning_Face = True
                else :
                    CVar.IsTurning_Face = False

                #CVar.nPos_Servo = nPos_X/10
```

```
            if CVar.IsTurning_Face == True :
                if (CVar.nPos_Servo> 512 + 200 ):
                    CVar.nPos_Servo = 512 + 200
                elif (CVar.nPos_Servo < 512 - 200 ):
                    CVar.nPos_Servo = 512 - 200
                #CVar.nPos_Servo = Filter_LowPass(0.3, CVar.nPos_Servo, nVal_Prev)

                OLLO( 1 , const.OLLO_JOINT_POSITION).write(CVar.nPos_Servo)
            else :
                if (m_IsFind == False ):
                    if motion.status() == False :
                        Motion_Play( 14 )
                m_IsFind = True


            #OLLO(1, const.OLLO_JOINT_POSITION).write(CVar.nPos_Servo)
            LedPwm( 0 , Value)
            #print(0, Value)
        else :
            Show_Text( 500 , 500 , 0 )
            LedPwm(Value, 0 )
            #print(Value, 0)
            #LedPwm(100 * int(Sin(CVar.m_nPwm/200 * 180)), 0)
            #LedPwm(CVar.m_nPwm * nLimit_Led / nTimeGap, 0)
            m_IsFind = False

'''
```

Page 33

```
def Page_Face_Track(nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0, Event_Up1,
  Btn0, Btn1):
    # Time reference value operated by pulse (ms)
    nTimeGap = 3000
    nTmr = tmrFace.Get()
    nDir = 1
    if (nTmr> nTimeGap * 2):
        nDir = 1
        nTmr = 0
        tmrFace.Set()
    elif (nTmr> nTimeGap):
        nDir = -1
    if (nDir> 0):
        CVar.m_nPwm = nTmr
    else:
        CVar.m_nPwm = nTimeGap * 2-nTmr
    # Brightness adjustment value of chest plate LED when searching face
    nLimit_Led = 100 # Max => 100 (0~100)
    if (rpi.area()> 0):
```

```
                nVal_Prev = CVar.nPos_Servo
                nCenterGap = 2
                Converted to step #10 (320 * 240)
                #The value that the robot looks to the left converges to 0
                nPos_X = rpi.position_x() * 10/320
                #Below y converges to 0
                nPos_Y = rpi.position_y() * 10/240
                nPos_Servo_Prev = CVar.nPos_Servo
                IsFind = False
                if (nPos_X <3-nCenterGap ):
                        CVar.nPos_Servo = CVar.nPos_Servo + 3
                        CVar.IsTurning_Face = True
                elif (nPos_X> 3 + nCenterGap ):
                        CVar.nPos_Servo = CVar.nPos_Servo-3
                        CVar.IsTurning_Face = True
                elif (CVar.IsTurning_Face == True):
                        CVar.IsTurning_Face = False
                        IsFind = True
                if (CVar.nPos_Servo> 512 + 200):
                        CVar.nPos_Servo = 512 + 200
                elif (CVar.nPos_Servo <512-200):
                        CVar.nPos_Servo = 512-200
                CVar.nPos_Servo = Filter_LowPass(0.5, CVar.nPos_Servo, nVal_Prev)
                OLLO(1, const.OLLO_JOINT_POSITION).write(CVar.nPos_Servo)
                LedPwm(0, CVar.m_nPwm * nLimit_Led / nTimeGap)
                if (IsFind == True):
                        Motion_Play_And_Wait(5)
        else:
                LedPwm(CVar.m_nPwm * nLimit_Led / nTimeGap, 0)
'''
def Page_MotorTest(nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0, Event_Up1,
  Btn0, Btn1):
```

Page 34

```
        if (Event_Dn0 == 1 ):
                nID = -1
                #Get the index of the first actuator.
                nFirstIndex = _BTN_ID_01[ 4 ]
                for i in range( 0 , 18 ):
                        nButtonIndex = i + nFirstIndex
                        if (nButtonIndex == nNum0) or (nButtonIndex == nNum1):
                                nID = i + 1
                if nID >= 0 :
                        Show_Motors(nID)
                        #All LEDs Off
                        DXL( 254 ).write8( 65 , 0 )
                        #LED on selected motor
                        DXL(nID).write8( 65 , 1 )
                        fRange = 10
                        # test moving
                        TestMove(nID, 0 , 30 )
```

```
                TestMove(nID, fRange, 30 )
                TestMove(nID, -fRange, 30 )
                TestMove(nID, 0 , 30 )
                Setup_Speed(nID, 0 )
def Page_Offset(nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0, Event_Up1,
  Btn0, Btn1):
    if (Event_Dn0 == 1 ):
        nID = -1
        #Get the index of the first actuator.
        nFirstIndex = _BTN_ID_01[ 4 ]
        for i in range( 0 , 18 ):
            nButtonIndex = i + nFirstIndex
            if (nButtonIndex == nNum0) or (nButtonIndex == nNum1):
                nID = i + 1
        if (nID >= 0 ):
            Show_Motors(nID)
            #All LEDs Off
            DXL( 254 ).write8( 65 , 0 )
            #LED on selected motor
            DXL(nID).write8( 65 , 1 )
            CVar.nID = nID
            CVar.nOffset = Offset_GetData(nID)
            ShowPage(_PAGE_OFFSET_DIALOG1)
            WaitButtonUp()
        else :
            CVar.nID = -1
            if (Btn0 == _BTN_RESET):
                ShowPage(_PAGE_OFFSET_DIALOG2)
                WaitButtonUp()
                #CVar.nOffset = 0
                CVar.IsResetCommand = True
            elif (Btn0 == _BTN_INIT):
                TorqAll( True , False )
                Move_Center( True )
```

Page 35

```
def Page_Offset_Dialog_1(nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0,
  Event_Up1, Btn0, Btn1):
    if (Btn0 == _BTN_DIALOG_PLUS): #+
        if (DXL(CVar.nID).read8( 64 ) == 0 ):
            #Torq Off -> On
            TorqOnOff(CVar.nID, True , False )
            Show_Image( 600 , 400 , 53 )
        CVar.nOffset = CVar.nOffset + 1
        etc.write16( 200 + CVar.nID * 2 , (int)(CVar.nOffset))
        DXL(CVar.nID).goal_position( 2048 + (int)(CVar.nOffset))
    if (Btn0 == _BTN_DIALOG_MINUS): #-
        if (DXL(CVar.nID).read8( 64 ) == 0 ):
            #Torq Off -> On
            TorqOnOff(CVar.nID, True , False )
```

```
                Show_Image( 600 , 400 , 53 )
                CVar.nOffset = CVar.nOffset-1
                etc.write16( 200 + CVar.nID * 2 , (int)(CVar.nOffset))
                DXL(CVar.nID).goal_position( 2048 + (int)(CVar.nOffset))
        if (CVar.nID> 0 ):
                Show_Num_Offset(CVar.nID)
        if (Event_Dn0 == 1 ):
                if (Btn0 == _BTN_DIALOG_OK):
                        if (DXL(CVar.nID).read8( 64 ) == 0 ):
                                CVar.nOffset = DXL(CVar.nID).goal_position()- 2048
                        # Put Offset data
                        etc.write16( 200 + CVar.nID * 2 , (int)(CVar.nOffset))
                        # Save
                        ShowPage(_PAGE_OFFSET_DIALOG2)
                        TorqAll( True , False )
                if (Btn0 == _BTN_DIALOG_CANCEL):
                        ShowPage(_PAGE_OFFSET)
                if (Btn0 == _BTN_DIALOG_TORQ):
                        if (CVar.nID> 0 ):
                                if (DXL(CVar.nID).read8( 64 ) == 0 ):
                                        #Torq Off -> On
                                        TorqOnOff(CVar.nID, True )
                                        Show_Image( 600 , 400 , 53 )
                                        CVar.nOffset = DXL(CVar.nID).goal_position()- 2048
                                        etc.write16( 200 + CVar.nID * 2 , (int)(CVar.nOffset))
                                else :
                                        #Torq On -> Off
                                        TorqOnOff(CVar.nID, False )
                                        Show_Image( 600 , 400 , 54 )
                                etc.write16( 200 + CVar.nID * 2 , (int)(CVar.nOffset))
def Page_Offset_Dialog_2(nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0,
  Event_Up1, Btn0, Btn1):
        global aOffset
        if (Event_Dn0 == 1 ):
                if (Btn0 == _BTN_DIALOG_OK):
                        if (CVar.IsResetCommand == True ):
                                Offset_Clear()
```



```
                        else :
                                Offset_Write()
                                aOffset[CVar.nID] = CVar.nOffset
                        # Done-Close the dialog
                        ShowPage(_PAGE_OFFSET)
                        Move_Center( True )
                if (Btn0 == _BTN_DIALOG_CANCEL):
                        ShowPage(_PAGE_OFFSET)
                        #ShowPage(_PAGE_OFFSET_DIALOG_CLOSE)

def Show_Dialog(nStep, nID= 5 ):
        # Show Offset Dialog
```

```
        if (nStep == 1 ):
                CVar.IsResetCommand = False
                # Show dialog background
                Show_Image( 500 , 500 , 51 )
                #Torq On/Off Image
                Show_Image( 600 , 400 , 53 )
                #DXL ID display
                Show_Num( 475 , 210 , nID, _COLOR_WHITE, 80 )
        #Show Save Dialog
        elif (nStep == 2 ):
                #Clear all numbers
                Clear_Num()
                # Show dialog background
                Show_Image( 500 , 500 , 52 )
                #DXL ID display
                Show_Num( 475 , 210 , nID, _COLOR_WHITE, 80 )
                #SAVE text output
                Show_Text( 500 , 500 , 11 ,_COLOR_BLACK, 100 )
        else :
                CVar.IsResetCommand = False
                #Clear-dialog background
                Show_Image( 500 , 500 , 0 )
                #Clear-Torq On/Off Image
                Show_Image( 600 , 400 , 0 )
                #Clear-Show Text "Save"
                Show_Text( 500 , 500 , 0 )
                #Clear all numbers
                Clear_Num()
                # If you close the window, do Torq On -> Move Center.
                TorqAll( True , False )
                if (CVar.nPage == _PAGE_OFFSET):
                        Move_Center( True )
                else :
                        Move_Center()

def Show_Num_Offset(nID):
        nOffset = DXL(nID).goal_position()- 2048
        nColor = 5
        nGap = 30
```



```
        if (nOffset < 0 ):
                nOffset = -nOffset
                nColor = 3
        nPos_X = 420
        nPos_Y = 560
        ### Clear
        nSize = 0
        for j in range( 0 , 4 ):
                if (nOffset <pow( 10 , (j + 1 ))):
                        for i in range( 0 , 4 -j):
```

```
                    Clear_Num(nPos_X + nGap * i, nPos_Y) # Set Show_Num Size to 0
                      It can be deleted by using, but for readability, make and use the Clear_Num() function.

                        break
        ### Display
        nSize = 120
        for i in range( 0 , 5 ):
                j = (int)(pow( 10 , ( 4 -i)))
                if (nOffset >= j) or (i == 4 ): The digit of # 1 (i == 4) should always be indicated if
                  It is tied in or condition so that it does not get caught in the door.
                        nValue = (int)(nOffset / j)% 10
                        Show_Num(nPos_X + nGap * i, nPos_Y, nValue, nColor, nSize)
def Move_Offset1(nID, fAngle):
        etc.write16( 264 + nID * 2 , CalcAngle2Raw(fAngle) -2048 )
def Move_Offset2(nID0, nRaw0, nID1, nRaw1):
        etc.write16( 264 + nID0 * 2 , nRaw0 -2048 )
        etc.write16( 264 + nID1 * 2 , nRaw1 -2048 )
def Defense(nMotion):
        nVal_Prev = 0
        pitch_knee_Prev = 0
        pitch_Up_Prev = 0
        pitch_Arm_Prev = 0
        nAction = 0
        Limit_Roll = 20
        Limit_Pitch = 20
        Sensing_Roll = 10
        Sensing_Pitch = 6

        nErrorCnt = 100
        while True :
                # Check button press
                nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0, Event_Up1, Btn0, Btn1 =
                  GetButton(btnList)
                #if CheckButton(Btn0, Btn1, _BTN_ACT_10) != 0:
                # break
                if nNum0 >= 0 or nNum1 >= 0 :
                        break
                #Motion_Ready()
                #Motion_Play(nMotion)
                motion.play(nMotion)
                while True :
```



```
                        if motion.status() == True :
                                roll = imu.roll()/ 100.0
                                pitch = imu.pitch()/ 100.0
                                nAction = 0

                                if (roll> Sensing_Roll):
                                        nAction = 57 # right-run in the direction of falling
```

```
elif (roll <-Sensing_Roll):
        nAction = 56 # left-run in the direction of falling


#if (roll> Sensing_Roll):
# nAction = 56 # left-run away from falling
#elif (roll <-Sensing_Roll):
# nAction = 57 # right-run away from falling


'''
if (pitch> Sensing_Pitch):
        if roll >= 0:
                nAction = 71#155
        else:
                nAction = 70
elif (pitch <-Sensing_Pitch):
        if roll >= 0:
                nAction = 73#160
        else:
                nAction = 72
'''

if nAction> 0 :
        Motion_Stop()
        #Motion_Wait()
        while motion.status():
                pass
        Motion_Play_And_Wait(nAction)
        #Offset_Clear()
        for i in range( 1 , 17 ):
                Move_Offset1(i, 0 )
        Motion_Play_And_Wait(nMotion)
        #Motion_Play(nMotion)
else :

        if (roll> Limit_Roll):
                roll = Limit_Roll
        elif (roll <-Limit_Roll):
                roll = -Limit_Roll
        if (pitch> Limit_Pitch):
                pitch = Limit_Pitch
        elif (pitch <-Limit_Pitch):
                pitch = -Limit_Pitch
```



```
        roll_Up = -roll * 2.0 #0.5
        weight_roll = 1.7 #1.2 # opposite leg
        weight_roll_Stand = 2.7 # 3 # Reference leg
        weight_roll_w = 2.0
```

```
Val_L = 2048
Val_R = 2048
if (roll_Up >= 0 ):
        Val_L = CalcAngle2Raw(roll_Up / weight_roll_Stand)
        Val_R = CalcAngle2Raw(roll_Up * weight_roll)
        Move_Offset1( 2 , roll_Up * weight_roll_w)
else :
        Val_L = CalcAngle2Raw(roll_Up * weight_roll)
        Val_R = CalcAngle2Raw(roll_Up / weight_roll_Stand)
        Move_Offset1( 4 , -roll_Up * weight_roll_w)
Move_Offset2( 5 , Val_R, 7 , Val_L)
Val2 = CalcAngle2Raw(roll_Up /
  3 ) #2048#CalcAngle2Raw(-roll_Up / 4)
Move_Offset2( 10 , Val2, 12 , Val2)
pitch_Up = pitch

#Hip joint
pitch_Up_Filter = pitch_Up * 3
if (pitch_Up_Prev == 0 ):
        pitch_Up_Prev = pitch_Up_Filter
pitch_Up_Filter = Filter_LowPass( 0.5 , pitch_Up_Filter,
  pitch_Up_Prev)
pitch_Up_Prev = pitch_Up_Filter
Val = CalcAngle2Raw(pitch_Up_Filter)
Val2 = CalcAngle2Raw(-pitch_Up_Filter)
Move_Offset2( 6 , Val, 8 , Val)
# Arm
nArm_Multi = 3
if abs(pitch_Up) < 5 :
        nArm_Multi = 4
pitch_Arm_Filter = pitch_Up * nArm_Multi # * 4
if (pitch_Arm_Prev == 0 ):
        pitch_Arm_Prev = pitch_Arm_Filter
pitch_Arm_Filter = Filter_LowPass( 0.6 , pitch_Arm_Filter,
  pitch_Arm_Prev)
pitch_Arm_Prev = pitch_Arm_Filter

Val = CalcAngle2Raw(pitch_Arm_Filter)
Val2 = CalcAngle2Raw(-pitch_Arm_Filter)
Move_Offset2( 1 , Val2, 3 , Val)
#Ankle
pitch_Dn_Filter = -pitch_Up* 2
if (nVal_Prev == 0 ):
        nVal_Prev = pitch_Dn_Filter
pitch_Dn_Filter = Filter_LowPass( 0.6 , pitch_Dn_Filter,
  nVal_Prev)
```



```
nVal_Prev = pitch_Dn_Filter
Val2 = CalcAngle2Raw(pitch_Dn_Filter)
```

```
Move_Offset2( 9 , Val2, 11 , Val2)
#print(Val2)

#knee
pitch_knee = pitch_Up* 1
if (pitch_knee_Prev == 0 ):
        pitch_knee_Prev = pitch_knee
pitch_knee = Filter_LowPass( 0.6 , pitch_knee,
  pitch_knee_Prev)
pitch_knee_Prev = pitch_knee
Val_Knee_0 = CalcAngle2Raw(pitch_knee)
Val_Knee_1 = CalcAngle2Raw(-pitch_knee)
Move_Offset2( 15 , Val_Knee_0, 16 , Val_Knee_1)

#nLimit_Defense_Front = 2120
nLimit_Defense_Front = 2150 -( 50 ) # -is insensitive + is sensitive
#nLimit_Defense_Back = 1930
nLimit_Defense_Back = 1900 +( -10 ) # -is insensitive + is sensitive
if (nErrorCnt> 0 ):
        nErrorCnt = nErrorCnt- 1
        nLimit_Defense_Back = nLimit_Defense_Back- 1000
        nLimit_Defense_Front = nLimit_Defense_Front + 1000
        #if (nErrorCnt == 0):
        # print(nErrorCnt)

nAction = 0
# If you fall forward, Val2 decreases (if you hit from the back)
#if (Val2 <1950):
if (Val2 <nLimit_Defense_Back): #1950
        nErrorCnt = 100
        Motion_Stop()
        #nAction = 250
        if roll >= 0 :
                nAction = 71 #401#71#155
        else :
                nAction = 70 #400#70
#elif (Val2> 2100):
#elif (Val2> 2120):

#elif (Val2> 2150):
elif (Val2> nLimit_Defense_Front): #2150
        nErrorCnt = 100
        Motion_Stop()
        if roll >= 0 :
                nAction = 73 #160
        else :
                nAction = 72
if nAction> 0 :
        #Motion_Wait()
```

```
                                        while motion.status():
                                            pass
                                        #Offset_Clear()
                                        Motion_Play_And_Wait(nAction)
                                        #delay(500)
                                        #Offset_Clear()
                                        #Motion_Play_And_Wait(nMotion)
                                        #delay(100)
                            else : # Motion End
                                    break

        buzzer.melody( 14 )
        #Offset_Clear()

        for i in range( 1 , 17 ):
                Move_Offset1(i, 0 )
################################## ######## ###############
  ###################
# Select the method of printing messages such as print from the following three.
#console(USB) # USB cable
#console(BLE) # bluetooth
console(UART) # LN101

#########################
# In actual use, nTest = 0 should be used.
# If you want to see the actual coordinates of the pressed place, set this value to 1 and touch it.
nTest = 0                                       # 0-Normal, 1-Coordinate output
nTest_BackgroundImage = 1                       # Page to be displayed during test
#########################

# Execute initial posture and settings only when not in test mode
if (nTest == 0 ):
        # controller direction: 0-vertical(Humanoid), 1-Horizontal
        eeprom.imu_type( 0 )
        #Sit down before you release the talk.
        Motion_Play_And_Wait( 60 )
        delay( 100 )
        # Turn off the torque and modify the actuator and controller settings to suit the robot.
        TorqAll( False )
        # profile -> velocity-based
        DXL( 254 ).write8( 10 , 0 ) # 0 -> velocity-based profile, 4 -> time-based
          profile
        # Secondary ID(255: No Use, 0 ~ 252: ID)
        DXL( 254 ).write8( 12 , 255 ) # 255 -> No Use(Clear)
        # Operation Mode(1:velocity[wheel], 3:position)
        DXL( 254 ).mode( 3 ) # position
        TorqAll( True )

        # Move your head to the center.
        OLLO( 1 , const.OLLO_JOINT_POSITION).write( 512 )
```

```
    # Return all DXL speeds to their initial state
    Setup_Speed( 254 , 0 )

    # Initial posture
    Motion_Ready( 1 )

    #Read All Offset
    Offset_Read()

while ( True ):
    if (IsPhone == False ):
        if (smart.is_connected() == True ):
            #Check if it is connected to the smartphone.
            smart.wait_connected()
            #Set the smartphone screen horizontally. (0: Auto, 1: Vertical, 2: Horizontal)
            smart.display.screen_orientation( 2 )
            # Waiting for the screen to switch before getting the screen's width and height.
            delay( 500 )
            # If there is a camera in action, close it.
            #0: Close, 1: Color, 2: Face Detection, 3: Streaming, 4: Marker, 5: Lane, 6: Emotion, 7: Hand
            rpi.mode( 0 )
            smart.write8( 10700 , 0 ) # Smartphone streaming video screen 0: Close, 1: Display
            # Get the resolution of the screen and put it in CVar.nScreenWidth, CVar.nScreenHeight.
            GetResolution()
            # Print background image
            ShowPage(_PAGE_REMOTE_NORMAL)
            # Record the smartphone connection in a variable
            IsPhone = True
    else :
        # Test 1 => Coordinate output
        if (nTest == 1 ):
            #Check the touch input of the smartphone
            GetTouch_Down()
            Test1(nTest_BackgroundImage)
        else : #Run
            #Check the bent state of the knee-to ensure that walking in a low posture is performed when it is over 100 degrees
              check
            CheckKnee()
            # Check button press
            nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0, Event_Up1, Btn0,
              Btn1 = GetButton(btnList)
            if (Event_Dn1 == 1 ):
                smart.etc.vibrate( 10 )
            elif (Event_Dn0 == 1 ):
                smart.etc.vibrate( 10 )

            #The menu bar is floating
            if (CVar.nPage == _PAGE_MENU):
                Page_MenuBar(nNum0, nNum1, Event_Dn0, Event_Dn1, Event_Up0,
                  Event_Up1, Btn0, Btn1)
            else : #If the menu bar is not floating
```

```
if (Btn0 == _BTN_MENU): # Press the menu button
        #if (CVar.nPage == _PAGE_STREAM):
        ShowPage(_PAGE_MENU) # Bring up the menu bar.
        WaitButtonUp()                          # Wait for the button to be released
else :
        if (
                (CVar.nPage == _PAGE_REMOTE_NORMAL) or
                (CVar.nPage == _PAGE_REMOTE_FIGHT) or
                (CVar.nPage == _PAGE_REMOTE_SPECIAL) or
                (CVar.nPage == _PAGE_STREAM)
                ):
                Page_Remote(nNum0, nNum1, Event_Dn0, Event_Dn1,
                  Event_Up0, Event_Up1, Btn0, Btn1)
        elif (CVar.nPage == _PAGE_FACE_TRACK):
                Page_Face_Track(nNum0, nNum1, Event_Dn0, Event_Dn1,
                  Event_Up0, Event_Up1, Btn0, Btn1)
        elif (CVar.nPage == _PAGE_MOTOR_TEST):
                Page_MotorTest(nNum0, nNum1, Event_Dn0, Event_Dn1,
                  Event_Up0, Event_Up1, Btn0, Btn1)
        elif (CVar.nPage == _PAGE_OFFSET):
                Page_Offset(nNum0, nNum1, Event_Dn0, Event_Dn1,
                  Event_Up0, Event_Up1, Btn0, Btn1)
        elif (CVar.nPage == _PAGE_OFFSET_DIALOG1):
                Page_Offset_Dialog_1(nNum0, nNum1, Event_Dn0,
                  Event_Dn1, Event_Up0, Event_Up1, Btn0, Btn1)

        elif (CVar.nPage == _PAGE_OFFSET_DIALOG2):
                Page_Offset_Dialog_2(nNum0, nNum1, Event_Dn0,
                  Event_Dn1, Event_Up0, Event_Up1, Btn0, Btn1)
```