МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМ. Р.Е. АЛЕКСЕЕВА

ИНСТИТУТ РАДИОЭЛЕКТРОНИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

|  |  |  |
| --- | --- | --- |
|  |  |  |

Курс “Аппаратное и программное обеспечение роботизированных систем”

Отчет по лабораторной работе №4

Выполнили:

Бубнов Е. А

Проверил: Гай В.Е.

Нижний Новгород 2022

**Задача 7.**  **Управление автомобилем (Humanoid Sprint)**

Этот тест направлен на разработку компьютерной программы, которая управляет гуманоидным роботом, чтобы идти как можно быстрее по 10-метровой дорожке. Язык программирования - Python, а модель гуманоидного робота - робот Aldebaran NAO.

**Описание работы:**

1. сначала происходит инициализация моторов:

def initialize(self):

        """Get device pointers, enable sensors and set robot initial pose."""

        # This is the time step (ms)

        self.timeStep = int(self.getBasicTimeStep())

        # Get pointers to the shoulder motors.

        self.RShoulderPitch = self.getMotor('RShoulderPitch')

        self.LShoulderPitch = self.getMotor('LShoulderPitch')

        # Get pointers to the 12 motors of the legs

        self.RHipYawPitch = self.getMotor('RHipYawPitch')

        self.LHipYawPitch = self.getMotor('LHipYawPitch')

        self.RHipRoll     = self.getMotor('RHipRoll')

        self.LHipRoll     = self.getMotor('LHipRoll')

        self.RHipPitch    = self.getMotor('RHipPitch')

        self.LHipPitch    = self.getMotor('LHipPitch')

        self.RKneePitch   = self.getMotor('RKneePitch')

        self.LKneePitch   = self.getMotor('LKneePitch')

        self.RAnklePitch  = self.getMotor('RAnklePitch')

        self.LAnklePitch  = self.getMotor('LAnklePitch')

        self.RAnkleRoll   = self.getMotor('RAnkleRoll')

        self.LAnkleRoll   = self.getMotor('LAnkleRoll')

1. правила для движения левой ноги:

# move the left foot (keep the foot paralell to the ground)

    def left(self, x, y, z):

        # x, z

        self.LKneePitch.setPosition ( z      )

        self.LHipPitch.setPosition  (-z/2 + x)

        self.LAnklePitch.setPosition(-z/2 - x)

        # y

        self.LHipRoll.setPosition  ( y)

        self.LAnkleRoll.setPosition(-y)

1. правила для движения правой ноги:

# move the right foot (keep the foot paralell to the ground)

    def right(self, x, y, z):

        # x, z

        self.RKneePitch.setPosition ( z      )

        self.RHipPitch.setPosition  (-z/2 + x)

        self.RAnklePitch.setPosition(-z/2 - x)

        # y

        self.RHipRoll.setPosition  ( y)

        self.RAnkleRoll.setPosition(-y)

1. запуск:

def run(self):

        # some parameters to play around. Uncomment the

        # parameter set to try it out.

        ## 1. slow and steady

        # f            = 4

        # robot\_height = 0.5

        # shift\_y      = 0.3

        # step\_height  = 0.4

        # step\_length  = 0.2

        # arm\_swing    = 2.0

        ## 2. larger steps and steady

        # f            = 10

        # robot\_height = 0.9

        # shift\_y      = 0.2

        # step\_height  = 0.5

        # step\_length  = 0.25

        # arm\_swing    = 2.0

        ## 3. run

        f            = 14

        robot\_height = 0.9

        shift\_y      = 0.15

        step\_height  = 0.7

        step\_length  = 0.4

        arm\_swing    = 1.5

        while self.step(self.timeStep) != -1:

            # scale the time to modulate the frequency of the walk

            t = self.getTime()\*f

            # y

            yLeftRight = math.sin(t)\*shift\_y

            # z

            zLeft  = (math.sin(t)          + 1.0) / 2.0 \* step\_height + robot\_height

            zRight = (math.sin(t + math.pi)+ 1.0) / 2.0 \* step\_height + robot\_height

            # x

            # math.sin(t + math.pi/2) = math.cos(t)

            xLeft  = math.cos(t          )\*step\_length

            xRight = math.cos(t + math.pi)\*step\_length

            # apply

            self.left(  xLeft, yLeftRight, zLeft )

            self.right(xRight, yLeftRight, zRight)

            # move shoulders to stabilize steps

            self.RShoulderPitch.setPosition(arm\_swing\*xLeft  + math.pi/2  -0.1)

            self.LShoulderPitch.setPosition(arm\_swing\*xRight + math.pi/2  -0.1)

*Есть несколько вариантов ходьбы робота:*

Медленно и стабильно - 1,07с

## 1. slow and steady

        # f            = 4

        # robot\_height = 0.5

        # shift\_y      = 0.3

        # step\_height  = 0.4

        # step\_length  = 0.2

        # arm\_swing    = 2.0

Большой шаг, но иногда уходит от прямой траектории(Средне) - 48с

## 2. larger steps and steady

        # f            = 10

        # robot\_height = 0.9

        # shift\_y      = 0.2

        # step\_height  = 0.5

        # step\_length  = 0.25

        # arm\_swing    = 2.0

Быстро, но большая вероятность, споткнуться и упасть: 27,60с

## 3. run

        f            = 14

        robot\_height = 0.9

        shift\_y      = 0.15

        step\_height  = 0.7

        step\_length  = 0.4

        arm\_swing    = 1.5

1. отрегулировали скорость в соответствии со значением, возвращаемым датчиком расстояния спереди:

controller = Sprinter()

controller.initialize()

controller.run()

**Приложение:**

# A simple open loop controller generating walking behavior

# on a humanoid robot NAO.

# We use the asumption of 'parallel kinematics', i.e., feet are

# kept parallel to the ground, to simplify kinematic calculations.

# The motion dynamics is generated with sin/cos based oscillations.

from controller import Robot

import math

class Sprinter(Robot):

    """Make the NAO robot run as fast as possible."""

    def initialize(self):

        """Get device pointers, enable sensors and set robot initial pose."""

        # This is the time step (ms)

        self.timeStep = int(self.getBasicTimeStep())

        # Get pointers to the shoulder motors.

        self.RShoulderPitch = self.getMotor('RShoulderPitch')

        self.LShoulderPitch = self.getMotor('LShoulderPitch')

        # Get pointers to the 12 motors of the legs

        self.RHipYawPitch = self.getMotor('RHipYawPitch')

        self.LHipYawPitch = self.getMotor('LHipYawPitch')

        self.RHipRoll     = self.getMotor('RHipRoll')

        self.LHipRoll     = self.getMotor('LHipRoll')

        self.RHipPitch    = self.getMotor('RHipPitch')

        self.LHipPitch    = self.getMotor('LHipPitch')

        self.RKneePitch   = self.getMotor('RKneePitch')

        self.LKneePitch   = self.getMotor('LKneePitch')

        self.RAnklePitch  = self.getMotor('RAnklePitch')

        self.LAnklePitch  = self.getMotor('LAnklePitch')

        self.RAnkleRoll   = self.getMotor('RAnkleRoll')

        self.LAnkleRoll   = self.getMotor('LAnkleRoll')

    # move the left foot (keep the foot paralell to the ground)

    def left(self, x, y, z):

        # x, z

        self.LKneePitch.setPosition ( z      )

        self.LHipPitch.setPosition  (-z/2 + x)

        self.LAnklePitch.setPosition(-z/2 - x)

        # y

        self.LHipRoll.setPosition  ( y)

        self.LAnkleRoll.setPosition(-y)

    # move the right foot (keep the foot paralell to the ground)

    def right(self, x, y, z):

        # x, z

        self.RKneePitch.setPosition ( z      )

        self.RHipPitch.setPosition  (-z/2 + x)

        self.RAnklePitch.setPosition(-z/2 - x)

        # y

        self.RHipRoll.setPosition  ( y)

        self.RAnkleRoll.setPosition(-y)

    def run(self):

        # some parameters to play around. Uncomment the

        # parameter set to try it out.

        ## 1. slow and steady

        # f            = 4

        # robot\_height = 0.5

        # shift\_y      = 0.3

        # step\_height  = 0.4

        # step\_length  = 0.2

        # arm\_swing    = 2.0

        ## 2. larger steps and steady

        # f            = 10

        # robot\_height = 0.9

        # shift\_y      = 0.2

        # step\_height  = 0.5

        # step\_length  = 0.25

        # arm\_swing    = 2.0

        ## 3. run

        f            = 14

        robot\_height = 0.9

        shift\_y      = 0.15

        step\_height  = 0.7

        step\_length  = 0.4

        arm\_swing    = 1.5

        while self.step(self.timeStep) != -1:

            # scale the time to modulate the frequency of the walk

            t = self.getTime()\*f

            # y

            yLeftRight = math.sin(t)\*shift\_y

            # z

            zLeft  = (math.sin(t)          + 1.0) / 2.0 \* step\_height + robot\_height

            zRight = (math.sin(t + math.pi)+ 1.0) / 2.0 \* step\_height + robot\_height

            # x

            # math.sin(t + math.pi/2) = math.cos(t)

            xLeft  = math.cos(t          )\*step\_length

            xRight = math.cos(t + math.pi)\*step\_length

            # apply

            self.left(  xLeft, yLeftRight, zLeft )

            self.right(xRight, yLeftRight, zRight)

            # move shoulders to stabilize steps

            self.RShoulderPitch.setPosition(arm\_swing\*xLeft  + math.pi/2  -0.1)

            self.LShoulderPitch.setPosition(arm\_swing\*xRight + math.pi/2  -0.1)

controller = Sprinter()

controller.initialize()

controller.run()

Результаты:





