

Thinking Fingers

Code Documentation V1.0

1 Namespace Documentation	2
1.1 Camera Namespace Reference	2
1.2 main Namespace Reference	2
1.3 RobotControl Namespace Reference	3
2 Class Documentation	3
2.1 Camera.Camera Class Reference	3
2.1.1 Detailed Description	4
2.1.2 Constructor & Destructor Documentation	4
2.1.3 Member Function Documentation	4
2.1.4 Member Data Documentation	7
2.2 RobotControl.RobotControl Class Reference	8
2.2.1 Detailed Description	10
2.2.2 Constructor & Destructor Documentation	10
2.2.3 Member Function Documentation	10
2.2.4 Member Data Documentation	15
2.3 main.RobotControl_Thread Class Reference	18
2.3.1 Detailed Description	19
2.3.2 Constructor & Destructor Documentation	19
2.3.3 Member Function Documentation	19

1 Namespace Documentation

1.1 Camera Namespace Reference

Classes

- class Camera
take a picture and analyse it to detect a dice

1.2 main Namespace Reference

Classes

- class RobotControl_Thread
Class RobotControl_Thread update position of the robot.

Functions

- def getData ()
launch robotController Thread
- def stateMachine (ev=int)
stateMachine manage the state of the soft : init, running, stop

Variables

- int state = STATE_INIT
actual state for the state machine
 - int oldState = STATE_INIT
old state for the state machine
 - theRobotController = RobotControl()
class RobotControl object
 - theCamera = Camera()
class Camera object
 - t = Timer(0.1,getData)
timer to launch thread periodically
-

1.3 RobotControl Namespace Reference

Classes

- class RobotControl
Control the robot.

2 Class Documentation

2.1 Camera.Camera Class Reference

take a picture and analyse it to detect a dice

Public Member Functions

- def __init__(self)
the constructor
- def initRelation (self, robotController)
intialise relation
- def capture (self)
capture an image
- def cameraDetectionDice (self)
manage the dice detection then trigger the state machine of class RobotControl depending on the dice number
- def detectNumberOnDice (self, image)
detect the number on a dice
- def midpoint (self, ptA, ptB)
calculate the mid point between 2 points
- def foundDice (self, imagePath, width_object)
search a dice in an image

Public Attributes

- deltaX_m
delta x of the object in meter from the center of the camera
 - deltaY_m
delta y of the object in meter from the center of the camera
 - angleRot
orientation the object in radian
 - pixelsPerMeter
pixels per meter ration
 - imgCrop
image of the dice
 - robotController
robotController object
 - camera
camera object
-

2.1.1 Detailed Description

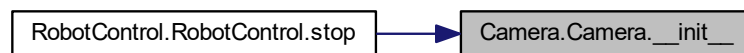
take a picture and analyse it to detect a dice

2.1.2 Constructor & Destructor Documentation

2.1.2.1 `__init__()` `def Camera.Camera.__init__ (`
`self)`

the constructor

Here is the caller graph for this function:



2.1.3 Member Function Documentation

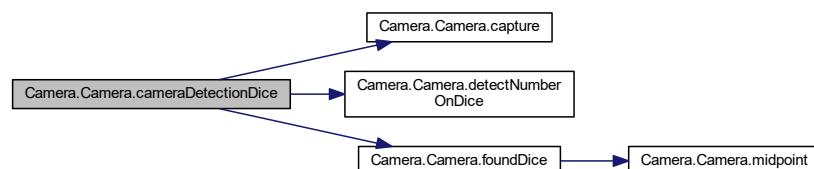
2.1.3.1 `cameraDetectionDice()` `def Camera.Camera.cameraDetectionDice (`
`self)`

manage the dice detection then trigger the state machine of class RobotControl depending on the dice number

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

Here is the call graph for this function:



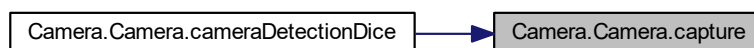
2.1.3.2 capture() `def Camera.Camera.capture (`
 `self)`

capture an image

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

Here is the caller graph for this function:



2.1.3.3 detectNumberOnDice() `def Camera.Camera.detectNumberOnDice (`
 `self,`
 `image)`

detect the number on a dice

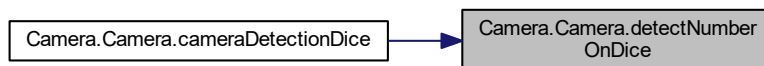
Parameters

<i>self</i>	The object pointer.
<i>image</i>	image of the dice

Returns

the dice number

Here is the caller graph for this function:



2.1.3.4 foundDice() `def Camera.Camera.foundDice (`
 `self,`
 `imagePath,`
 `width_object)`

search a dice in an image

Parameters

<i>self</i>	The object pointer.
<i>imagePath</i>	path to the image
<i>width_object</i>	width in mm of the object

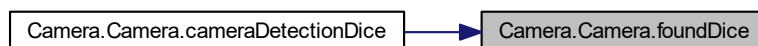
Returns

an image of the dice

Here is the call graph for this function:



Here is the caller graph for this function:



2.1.3.5 initRelation() `def Camera.Camera.initRelation (`
 `self,`
 `robotController)`

initialise relation

Parameters

<i>self</i>	The object pointer.
<i>self</i>	The robot controller object

2.1.3.6 midpoint() `def Camera.Camera.midpoint (`
 `self,`
 `ptA,`
 `ptB)`

calculate the mid point between 2 points

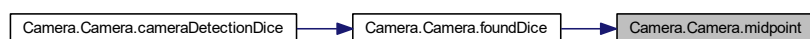
Parameters

<i>self</i>	The object pointer.
<i>ptA</i>	point 1
<i>ptB</i>	point 2

Returns

the middle point

Here is the caller graph for this function:



2.1.4 Member Data Documentation

2.1.4.1 angleRot `Camera.Camera.angleRot`

orientation the object in radian

2.1.4.2 camera `Camera.Camera.camera`

camera object

2.1.4.3 deltaX_m `Camera.Camera.deltaX_m`

delta x of the object in meter from the center of the camera

2.1.4.4 deltaY_m `Camera.Camera.deltaY_m`

delta y of the object in meter from the center of the camera

2.1.4.5 imgCrop `Camera.Camera.imgCrop`

image of the dice

2.1.4.6 robotController `Camera.Camera.robotController`

robotController object

2.2 RobotControl.RobotControl Class Reference

Control the robot.

Public Member Functions

- `def __init__ (self)`
constructor
 - `def initRelations (self, theCamera)`
get camera object
 - `def calibrate (self)`
Put the robot in original state.
 - `def updateCurrentPosition (self)`
update robot position get position X,Y,Z and orientation of tool center point
 - `def moveToPosition (self, x=float, y=float, z=float, rz=float)`
move the robot to a position XYZ with angle rz
 - `def setObjectPosition (self, dx=float, dy=float, rz=float)`
set the object found position
 - `def statePliers (self)`
check the state of the pliers
 - `def adjustPliers (self, pliersState)`
open or close the pliers
 - `def master (self, event)`
State machine who manage the soft as follow : Step 1 : Move the pliers Step 2 : Search the dice Step 3 : dice found --> go step 4, else step 1 Step 4 : Grab the dice Step 5 : Launch the dice and go back to step 1 Stop if the dice is 6.
 - `def stop (self)`
this function stop the robot and put him in original state
-

Public Attributes

- angularvelocity
angular velocity of the robot
 - angularacceleration
angular acceleration of the robot
 - linearvelocity
linear velocity of the robot
 - linearacceleration
linear acceleration of the robot
 - posx
tool center point position x
 - posy
tool center point position y
 - posz
tool center point position z
 - rx
tool center point orientation rx
 - ry
tool center point orientation ry
 - rz
tool center point orientation rz
 - object_posX
x position of the object to catch
 - object_posY
y position of the object to catch
 - object_Rz
 - takeOrRelease
 - ZeroReached
minimal y value
 - MaxReached
maximal y value
 - evZone
zone to search the dice
 - lastZone
last zone where the dice has been search
 - BorderReached
border x value
 - xSearch
position x to search the object
 - ySearch
position y to search the object
 - zSearch
position z to search the object
 - host
IP Address of the robot.
 - state
state of the state machine
 - oldState
oldstate of the state machine
 - theCamera
camera object
 - object_posZ
-

2.2.1 Detailed Description

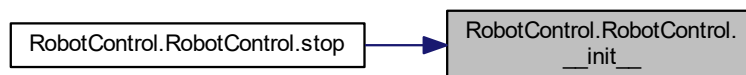
Control the robot.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 `__init__()` `def RobotControl.RobotControl.__init__ (`
`self)`

constructor

Here is the caller graph for this function:



2.2.3 Member Function Documentation

2.2.3.1 `adjustPliers()` `def RobotControl.RobotControl.adjustPliers (`
`self,`
`pliersState)`

open or close the pliers

Parameters

<i>pliersState</i>	True->Open, False->Close
<i>self</i>	The object pointer.

2.2.3.2 calibrate() `def RobotControl.RobotControl.calibrate (`
 self)

Put the robot in original state.

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

Here is the caller graph for this function:



2.2.3.3 initRelations() `def RobotControl.RobotControl.initRelations (`
 self,
 theCamera)

get camera object

Parameters

<i>self</i>	The object pointer.
<i>theCamera</i>	the camera object.

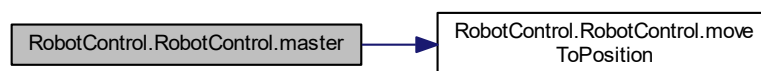
2.2.3.4 master() `def RobotControl.RobotControl.master (`
 `self,`
 `event)`

State machine who manage the soft as follow : Step 1 : Move the pliers Step 2 : Search the dice Step 3 : dice found --> go step 4, else step 1 Step 4 : Grab the dice Step 5 : Launch the dice and go back to step 1 Stop if the dice is 6.

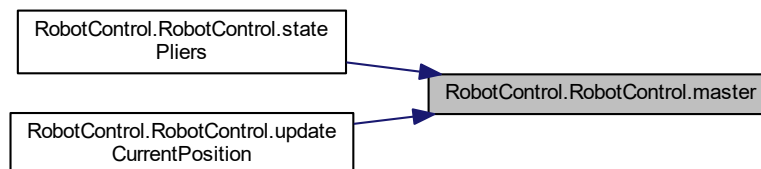
Parameters

<i>self</i>	The object pointer.
<i>event</i>	event to trigger state machine.

Here is the call graph for this function:



Here is the caller graph for this function:



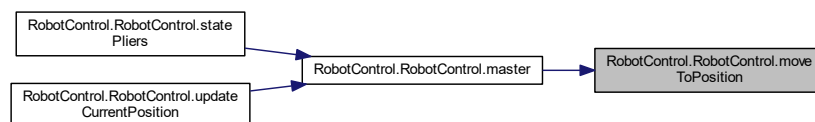
2.2.3.5 moveToPosition() `def RobotControl.RobotControl.moveToPosition (`
 `self,`
 `x = float,`
 `y = float,`
 `z = float,`
 `rz = float)`

move the robot to a position XYZ with angle rz

Parameters

<i>x</i>	: Position X of the Tool Center Point
<i>y</i>	: Position Y of the Tool Center Point
<i>z</i>	: Position Z of the Tool Center Point
<i>rz</i>	: orientation of the object
<i>self</i>	The object pointer.

Here is the caller graph for this function:



2.2.3.6 setObjectPosition() `def RobotControl.RobotControl.setObjectPosition (`
 `self,`
 `dx = float,`
 `dy = float,`
 `rz = float)`

set the object found postion

Parameters

<i>dx</i>	: Delta X of the object from the center of the camera
<i>dy</i>	: Delta Y of the object from the center of the camera
<i>rz</i>	: orientation of the object
<i>self</i>	The object pointer.

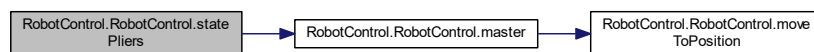
2.2.3.7 statePliers() `def RobotControl.RobotControl.statePliers (self)`

check the state of the pliers

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

Here is the call graph for this function:



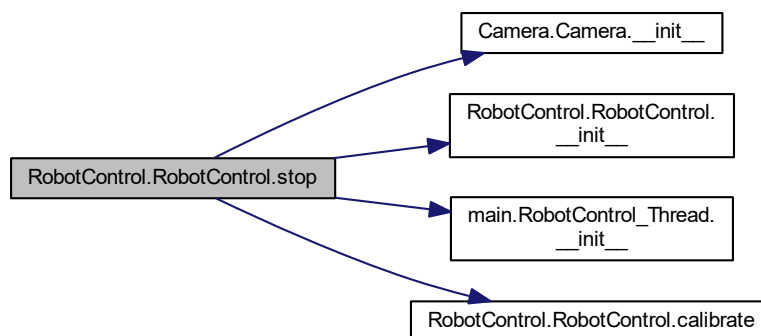
2.2.3.8 stop() `def RobotControl.RobotControl.stop (self)`

this function stop the robot and put him in original state

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

Here is the call graph for this function:



2.2.3.9 updateCurrentPosition() `def RobotControl.RobotControl.updateCurrentPosition (self)`

update robot position get position X,Y,Z and orientation of tool center point

Parameters

<i>self</i>	The object pointer.
-------------	---------------------

Here is the call graph for this function:



2.2.4 Member Data Documentation

2.2.4.1 angularacceleration `RobotControl.RobotControl.angularacceleration`

angular acceleration of the robot

2.2.4.2 angularvelocity `RobotControl.RobotControl.angularvelocity`

angular velocity of the robot

2.2.4.3 BorderReached `RobotControl.RobotControl.BorderReached`

border x value

2.2.4.4 evZone `RobotControl.RobotControl.evZone`

zone to search the dice

2.2.4.5 host `RobotControl.RobotControl.host`

IP Address of the robot.

2.2.4.6 lastZone `RobotControl.RobotControl.lastZone`

last zone where the dice has been search

2.2.4.7 linearacceleration `RobotControl.RobotControl.linearacceleration`

linear acceleration of the robot

2.2.4.8 linearvelocity `RobotControl.RobotControl.linearvelocity`

linear velocity of the robot

2.2.4.9 MaxReached `RobotControl.RobotControl.MaxReached`

maximal y value

2.2.4.10 object_posX `RobotControl.RobotControl.object_posX`

x postion of the object to catch

2.2.4.11 object_posY `RobotControl.RobotControl.object_posY`

y postion of the object to catch

2.2.4.12 object_posZ `RobotControl.RobotControl.object_posZ`**2.2.4.13 object_Rz** `RobotControl.RobotControl.object_Rz`**2.2.4.14 oldState** `RobotControl.RobotControl.oldState`

oldstate of the state machine

2.2.4.15 posx `RobotControl.RobotControl.posx`

tool center point position x

2.2.4.16 posy RobotControl.RobotControl.posy

tool center point position y

2.2.4.17 posz RobotControl.RobotControl.posz

tool center point position z

2.2.4.18 rx RobotControl.RobotControl.rx

tool center point orientation rx

2.2.4.19 ry RobotControl.RobotControl.ry

tool center point orientation ry

2.2.4.20 rz RobotControl.RobotControl.rz

tool center point orientation rz

2.2.4.21 state RobotControl.RobotControl.state

state of the state machine

2.2.4.22 takeOrRelease RobotControl.RobotControl.takeOrRelease**2.2.4.23 theCamera** RobotControl.RobotControl.theCamera

camera object

2.2.4.24 xSearch RobotControl.RobotControl.xSearch

position x to search the object

2.2.4.25 ySearch RobotControl.RobotControl.ySearch

position y to search the object

2.2.4.26 ZeroReached `RobotControl.RobotControl.ZeroReached`

minimal y value

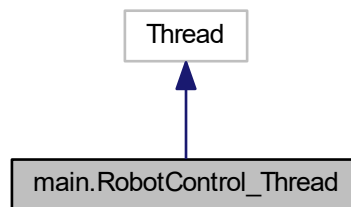
2.2.4.27 zSearch `RobotControl.RobotControl.zSearch`

position z to search the object

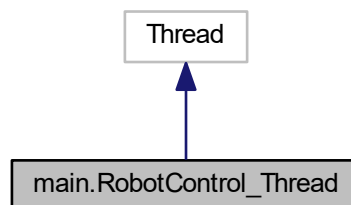
2.3 main.RobotControl_Thread Class Reference

Class `RobotControl_Thread` update position of the robot.

Inheritance diagram for `main.RobotControl_Thread`:



Collaboration diagram for `main.RobotControl_Thread`:

**Public Member Functions**

- `def __init__ (self)`
the constructor
 - `def run (self)`
get position of the robot
-

2.3.1 Detailed Description

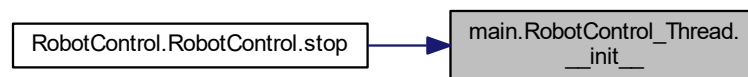
Class RobotControl_Thread update postion of the robot.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 __init__() `def main.RobotControl_Thread.__init__ (
 self)`

the constructor

Here is the caller graph for this function:



2.3.3 Member Function Documentation

2.3.3.1 run() `def main.RobotControl_Thread.run (
 self)`

get position of the robot

