UR5 API Documentation

# Livrables

Unity package avec une scène d’exemple.

La classe UR\_Manager gère la communication avec le robot.

Une fois connecté, sont disponibles les informations en temps réel (125hz) du robot.

Les commandes envoyées sont des programmes de commandes ur\_script. D’autres commandes sont disponibles (voir script manual).

# Fonctions

## Déplacement du robot

Id : D1

Contraintes : Spécifier une consigne de position cartésienne du centre de l’outil

Priorité : 1

Fonction : MoveL

/// <summary>

/// Moves the tcp to a position in the coordinates system of the robot

/// </summary>

/// <param name="poseT">translation vector in meters</param>

/// <param name="poseR">rotation vector in radians</param>

/// <param name="a">acceleration in m/s²</param>

/// <param name="v">speed in m/s</param>

/// <param name="t">time to reach in s</param>

/// <param name="r">bend radius in meters</param>

public void MoveL( Vector3 poseT, Vector3 poseR, double a = 1.2, double v = 0.25, double t = 0, double r = 0 )

Id : D2

Contraintes : Spécifier une consigne de vitesse cartésienne du centre de l’outil

Priorité : 1

Fonction : SpeedL

/// <summary>

/// Accelerate linearly in Cartesian space and continue with constant tool speed

/// </summary>

/// <param name="tSpeed">spatial vector</param>

/// <param name="rSpeed">rotation vector</param>

/// <param name="acceleration">tool linear acceleration</param>

/// <param name="minTime">time staying in the function</param>

/// <param name="rotAcc">tool angular acceleration</param>

public void SpeedL( Vector3 tSpeed, Vector3 rSpeed, float acceleration = 0.5f, float minTime = 0.1f, float rotAcc = 0.3f )

Id : D3 – D4 :

Contraintes : consignes en temps réel

Possibles en vitesse. En position, chaque nouvelle consigne commence par un stop.

Id : D5 :

Contraintes : Récupérer un retour si une consigne est invalide

Le robot ne permet pas cette fonction.

Id : D6

Contraintes : Activer et désactiver les freins de chaque articulation

Le robot ne permet pas cette fonction.

## Sécurité

Le robot ne permet pas de modifier les paramètres de sécurité internes.

## Outil

Id : O1

Contraintes : Spécifier le décalage entre l’extrémité du robot et le centre de l’outil

Fonction : SetTCPPosition

/// <summary>

/// Sets the tcp position in the coordinates system of the robot

/// </summary>

/// <param name="tcp\_vector">translation vector in meters from the toolmount</param>

/// <param name="rotation\_Vector">rotation vector in radians</param>

public void SetTCPPosition(Vector3 tcp\_vector, Vector3 rotation\_Vector)

Id : O2 – O3

Contraintes : spécifier la masse de l’outil et la position du centre de masse de l’outil

Fonction : SetPayload

/// <summary>

/// Sets the payload mass and center of gravity

/// </summary>

/// <param name="mass">mass in kilograms</param>

/// <param name="cog">center of gravity in meters from the toolmount</param>

public void SetPayload(double mass, Vector3 cog )

Id : O4

Contraintes : Récupérer la position cartésienne réelle du centre de l’outil

Fonction : GetTCPPosition

/// <summary>

/// Gets the actual cartesian coordinates of the tcp

/// </summary>

/// <returns> (x,y,z,rx,ry,rz) where rx,ry, rz is a rotation vector representation of the orientation</returns>

public double[] GetTCPPosition( )

Id : O5

Contraintes : Récupérer la vitesse cartésienne du centre de l’outil

Fonction : GetTCPSpeed

/// <summary>

/// Gets the actual speed of the of the tcp in cartesian coordinates

/// </summary>

/// <returns> (xd,yd,zd,rxd,ryd,rzd) where rxd,ryd, rzd is a rotation vector representation of the orientation</returns>

public double[] GetTCPSpeed( )

Id : O6

Contraintes : Récupérer les forces appliquées au centre de l’outil

Fonction : GetTCPForce

/// <summary>

/// Gets the generalised forces in the tcp

/// </summary>

/// <returns> (Fx,Fy,Fz,Mx,My,Mz) </returns>

public double[] GetTCPForce()

Id : O7

Contraintes : Récupérer les valeurs des entrées analogiques et/ou numériques du porte-outil

Fonctions : GetInputBits / GetAnalogInputs

/// <summary>

/// Gets the digital inputs

/// </summary>

/// <returns>Int64 where each bit is a digtal input</returns>

public Int64 GetInputBits()

/// <summary>

/// Gets the analog inputs

/// </summary>

/// <returns>Array of 4 analog inputs</returns>

public double[] GetAnalogInputs( )

Id : O8

Contraintes : spécifier la tension d’alimentation du connecteur présent dans le porte-outil

Fonction : SetVOutput

/// <summary>

/// enum of IO output voltage available

/// </summary>

public enum V\_OUTPUT

{

V\_0,

V\_12,

V\_24

}

/// <summary>

/// Sets the IO output voltage of the robot

/// </summary>

/// <param name="v">output voltage to set</param>

public void SetVOutput(V\_OUTPUT v)

## Etat du robot

Id : E1

Contraintes : Récupérer la position articulaire de chaque articulation

Fonction : GetJointPositions

/// <summary>

/// Gets the joint positions

/// </summary>

/// <returns>Array of six angular values in radians</returns>

public double[] GetJointPositions( )

Id : E2

Contraintes : Récupérer la vitesse articulaire de chaque articulation

Fonction : GetJointSpeeds

/// <summary>

/// Gets the joint speeds

/// </summary>

/// <returns>Array of six angular speed values in radians per second</returns>

public double[] GetJointSpeeds( )

# Fonctions supplémentaires

## Connection

/// <summary>

/// Starts the connection with the robot

/// </summary>

/// <param name="ip">IP adress of the robot</param>

public void Connect( string ip )

/// <summary>

/// Disconnect sockets to UR

/// </summary>

public void Disconnect( )

## Commandes

/// <summary>

/// Stops the robot in linear way with an acceleration of 1 m/s²

/// </summary>

public void StopL()

## Transformation de repère

Les repères cartésiens du robot et de Unity sont différents. Les fonctions suivantes permettent la transformation des « pose » en vecteur/quaternion et vice versa

/// <summary>

/// Transform a vector from robot to unity coordinate system

/// </summary>

/// <param name="poseT">vector from robot</param>

/// <returns></returns>

public Vector3 PoseTToVector3(Vector3 poseT)

/// <summary>

/// Transform a robot rotation vector from robot to unity quaternion coordinate system

/// </summary>

/// <param name="poseR">rotation vector</param>

/// <returns></returns>

public Quaternion PoseRToQuaternion( Vector3 poseR )

/// <summary>

/// Transform a unity vector to robot coordinate system

/// </summary>

/// <param name="v"></param>

/// <returns></returns>

public Vector3 Vector3ToPoseT( Vector3 v )

/// <summary>

/// Transform a quaternion from unity to robot rotation vector

/// </summary>

/// <param name="q">unity quaternion</param>

/// <returns></returns>

public Vector3 QuaternionToPoseR(Quaternion q )