**Theta1**

========== FUNCION Theta1.Update() ==========

SI theta = 0 ENTONCES

x <-- 0

SI NO ENTONCES

x <-- 360

FIN SI

**Theta3**

========== FUNCION Theta3.Update ==========

SI AngleDir(...) < 0 O DHParameters.getTheta2() > 90 ENTONCES

theta <-- theta \* -1

FIN SI

========== FUNCION Theta3.AngleDir ==========

SI dir > 0 ENTONCES

RETORNA 1

SI NO ENTONCES

SI dir < 0 ENTONCES

RETORNA -1

SI NO ENTONCES

RETORNA 0

FIN SI

FIN SI

**LowerArmSlider**

========== FUNCION LowerArmSlider.Start ==========

SI RobotBase.rotation.eulerAngles.y != 0 ENTONCES

SI (AngleDir(...) < 0) Y (RobotBase.rotation.eulerAngles.y > 180) ENTONCES

theta\*=-1

SI NO ENTONCES

SI (AngleDir(...) >= 0) Y (RobotBase.rotation.eulerAngles.y < 180) ENTONCES

theta\*=-1

FIN SI

FIN SI

========== FUNCION LowerArmSlider.Update ==========

SI DHParameters.getMoveSlider() ES VERDADERO ENTONCES

sliderTheta2.value <-- DHParameters.getTheta2()

lastTheta <-- DHParameters.getTheta2()

FIN SI

========== FUNCION LowerArmSlider.AngleDir ==========

SI dir > 0 ENTONCES

RETORNA 1

SI NO ENTONCES

SI dir < 0 ENTONCES

RETORNA -1

SI NO ENTONCES

RETORNA 0

FIN SI

FIN SI

========== FUNCION LowerArmSlider.SliderJoint1 ==========

SI DHParameters.getMoveSlider() ES FALSO false ENTONCES

RobotLowerArm.Rotate (-(angle - lastTheta), 0f , 0f)

lastTheta <-- angle

FIN SI

**LimitNeedle**

========== FUNCION LimitNeedle.OnTriggerEnter ==========

SI other.gameObject.name = "Ground" ENTONCES

savebtn.interactable <-- false;

FIN SI

========== FUNCION LimitNeedle.OnTriggerStay ==========

SI other.gameObject.name = "Ground" ENTONCES

savebtn.interactable <-- false

FIN SI

========== FUNCION LimitNeedle.OnTriggerExit ==========

SI other.gameObject.name = "Ground" ENTONCES

savebtn.interactable <-- true

FIN SI

**KnotPoints**

========== FUNCION KnotPoints.Start ==========

PARA i <-- 0 HASTA checks.Length CON PASO 1 HACER

checks[i].SetActive(false)

FIN PARA

========== FUNCION KnotPoints.SavePoint ==========

SI knotPoints < 5 ENTONCES

theta1Array[knotPoints] <-- BaseRotation.localRotation;

theta2Array[knotPoints] <-- Link1Rotation.localRotation;

theta3Array[knotPoints] <-- Link2Rotation.localRotation;

positionsArm[knotPoints] <-- jointPos.transform.position;

checks[knotPoints].SetActive(true);

knotPoints++;

FIN SI

========== FUNCION KnotPoints.ClearPoints ==========

PARA i <-- 0 HASTA checks.Length CON PASO 1 HACER

checks[i].SetActive(false)

FIN PARA

========== FUNCION KnotPoints.GoThroughPoints ==========

SI time.text != null ENTONCES

StartCoroutine(RotateMe(float.Parse(time.text)))

FIN SI

========== FUNCION KnotPoints.RotateMe ==========

PARA i <-- 0 HASTA knotPoints CON PASO 1 HACER

Quaternion baseFromAngle <-- BaseRotation.localRotation

Quaternion link1FromAngle <-- Link1Rotation.localRotation

Quaternion link2FromAngle <-- Link2Rotation.localRotation

PARA t <-- 0 HASTA < 1 CON PASO Time.deltaTime/inTime HACER

BaseRotation.localRotation <-- Quaternion.Lerp(...);

Link1Rotation.localRotation <-- Quaternion.Lerp(...);

Link2Rotation.localRotation <-- Quaternion.Lerp(...);

yield return null

FIN PARA

FIN PARA

========== FUNCION KnotPoints.DownloadCodeMELFA ==========

SI checks.Length > 0 ENTONCES

...

PARA i <-- 0 HASTA knotPoints CON PASO 1 HACER

positionAux <-- positionsArm[i];

sw.WriteLine(...);

FIN PARA

FIN SI

**UpperArmSlider**

========== FUNCION UpperArmSlider.Start ==========

SI AngleDir(...) < 0 ENTONCES

theta\*=-1

FIN SI

========== FUNCION UpperArmSlider.Update ==========

SI DHParameters.getMoveSlider() == TRUE ENTONCES

sliderTheta3.value <-- DHParameters.getTheta3()

lastTheta <-- DHParameters.getTheta3()

FIN SI

========== FUNCION UpperArmSlider.AngleDir ==========

SI dir > 0 ENTONCES

RETORNA 1

SI NO ENTONCES

SI dir < 0 ENTONCES

RETORNA -1

SI NO ENTONCES

RETORNA 0

FIN SI

FIN SI

========== FUNCION UpperArmSlider.SliderJoint2 ==========

SI DHParameters.getMoveSlider() = false ENTONCES

RobotUpperArm.Rotate (-(angle - lastTheta), 0f , 0f)

lastTheta <-- angle

FIN SI