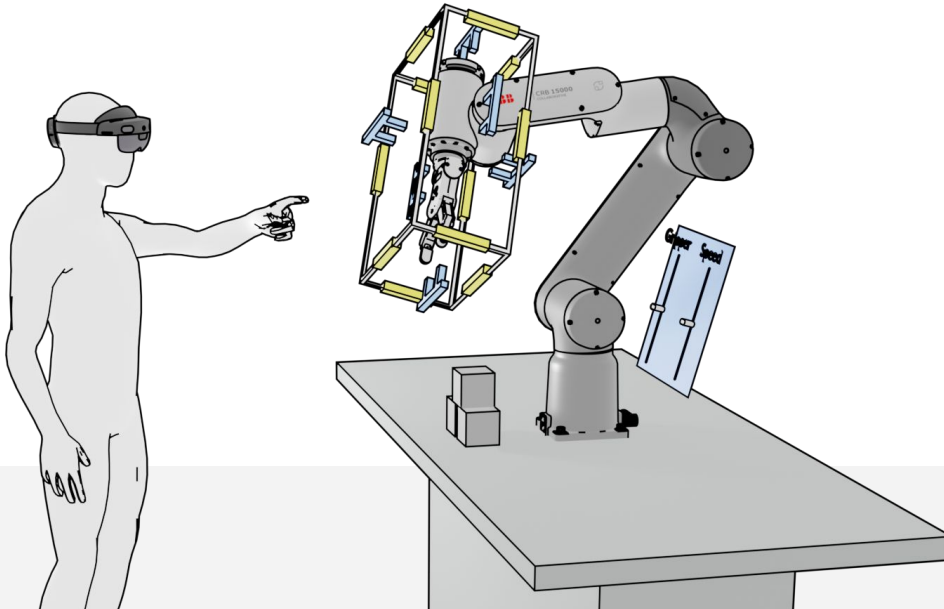


Externally Guided Motion (EGM)

Preparing your ABB robot

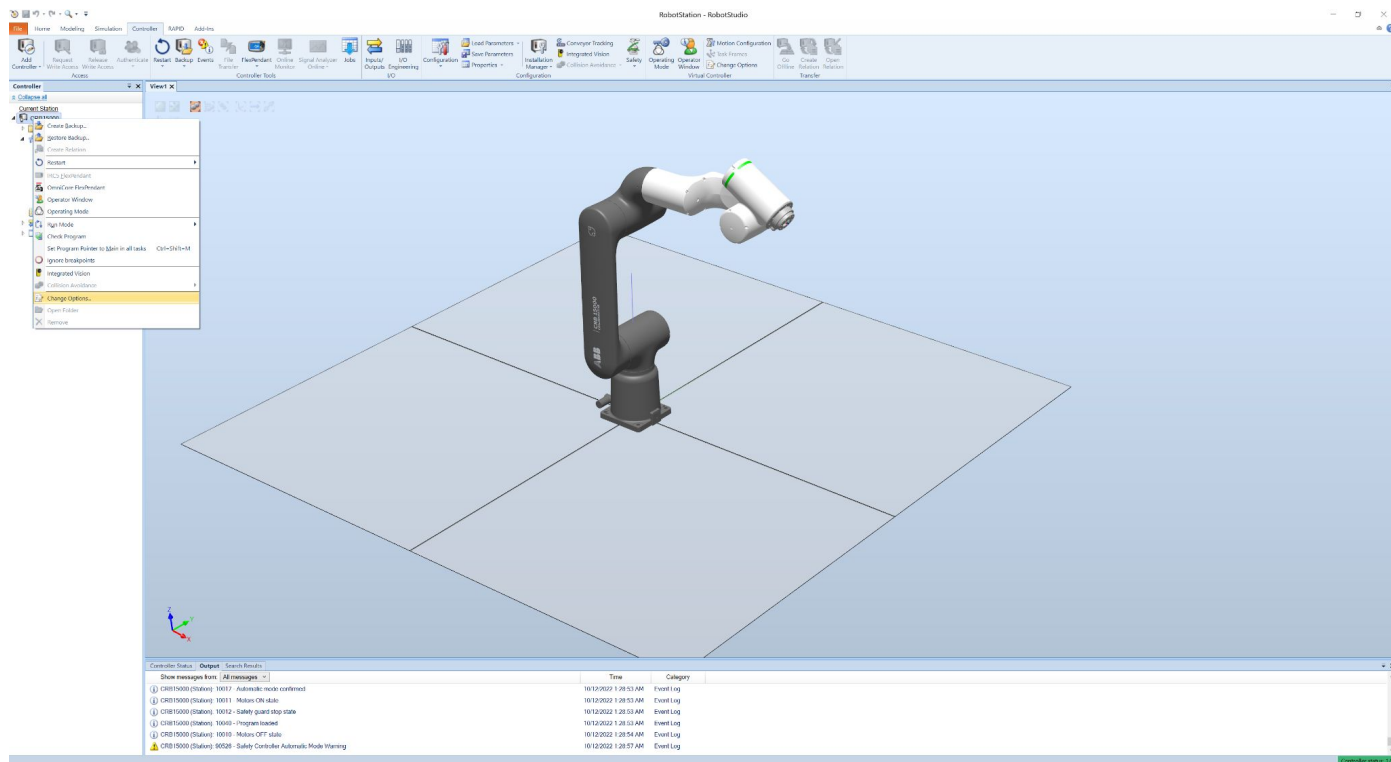


Disclaimer: This document is not sponsored or approved by ABB.

ABB updates EGM in every new version of RobotWare. Please always refer to their EGM application manual for more information.

Step 1: Enabling EGM on your controller

EGM is not enabled by default, you must enable it in the options of your robot controller:

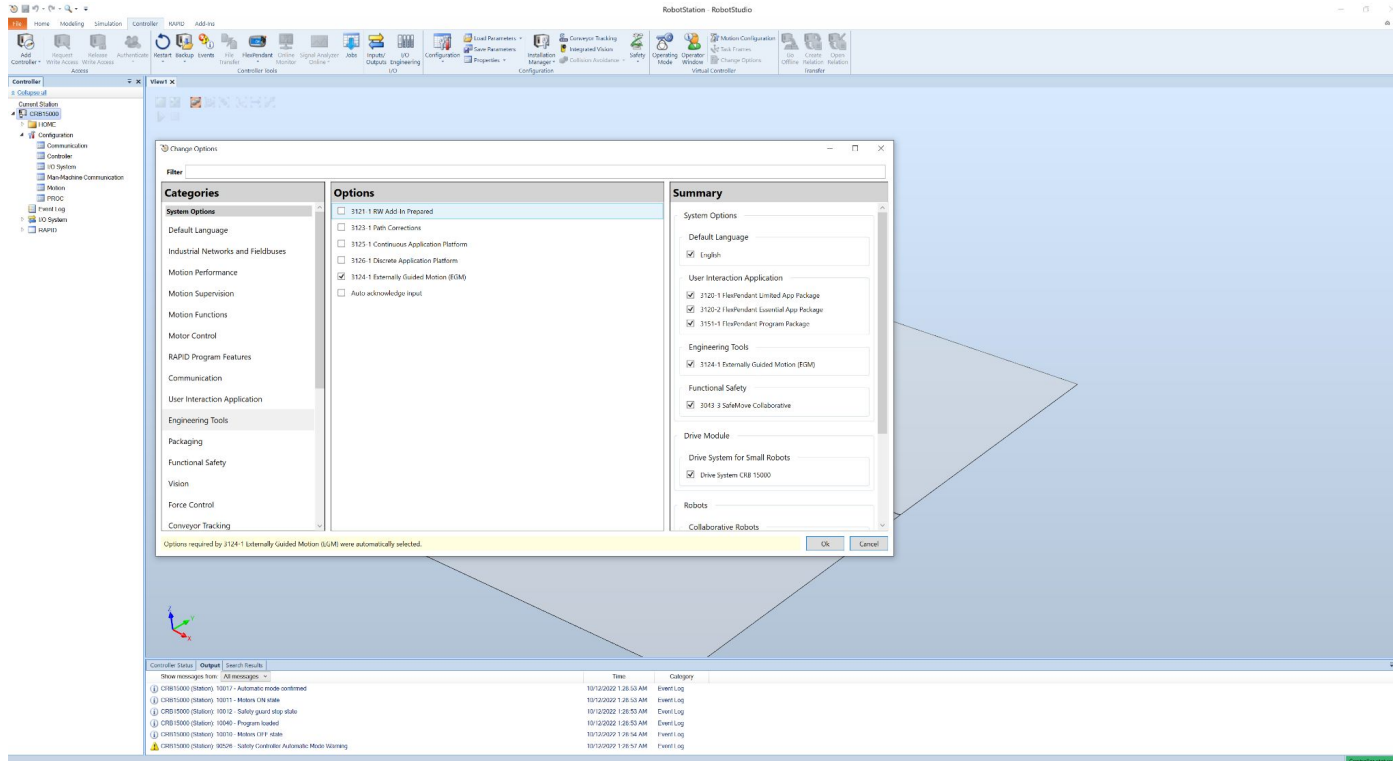


With RobotStudio opened, **right click** on your robot controller in the **Controller** tab, and click on the **Change Options...** button.

Make sure your computer is connected to a robot controller.

Step 1: Enabling EGM on your controller

EGM is not enabled by default, you must enable it in the system options of your robot controller:

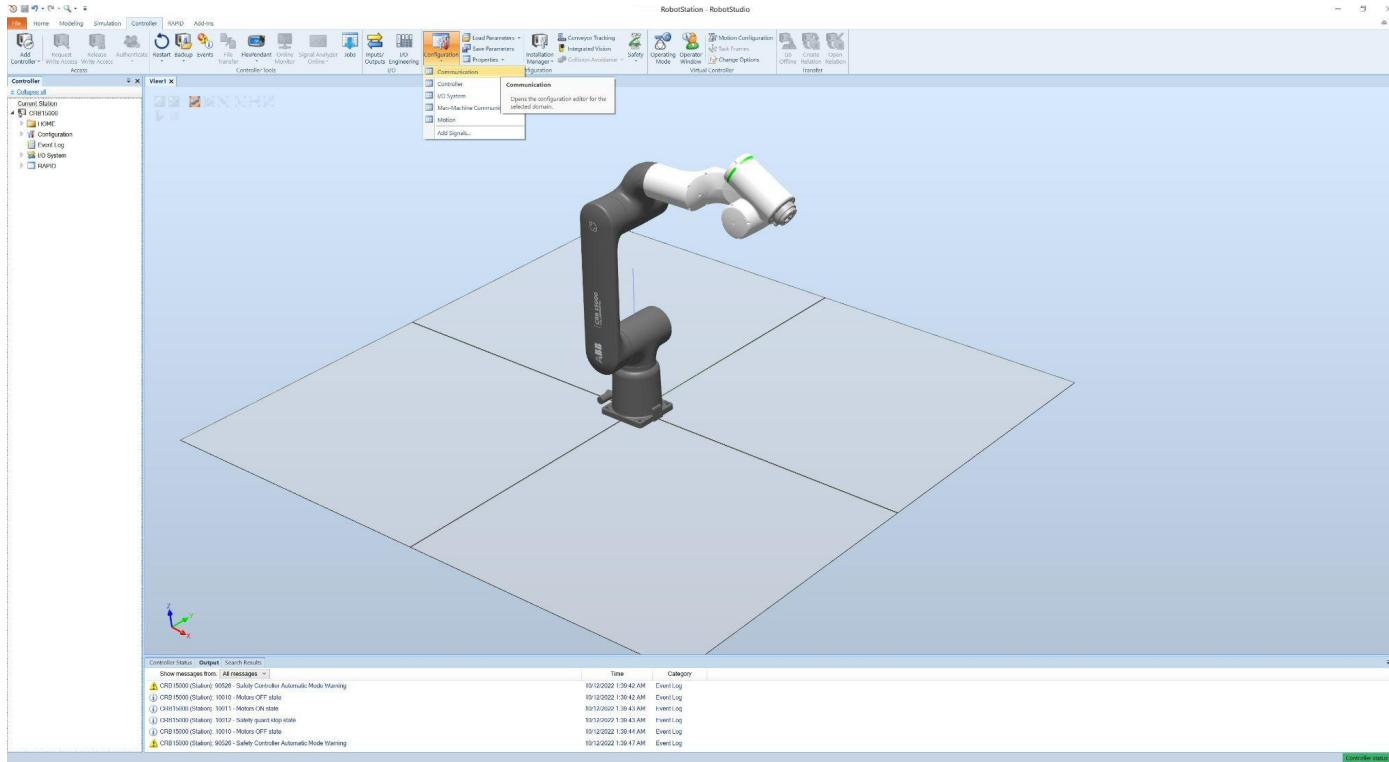


Find **Engineering Tools** in the system options menu, and select the **EGM** checkbox. Click on the **OK** button to confirm the change.

Older versions of RobotWare might not have EGM available by default. Please always contact ABB support for more information.

Step 2: Listening to a UDP Unicast Device

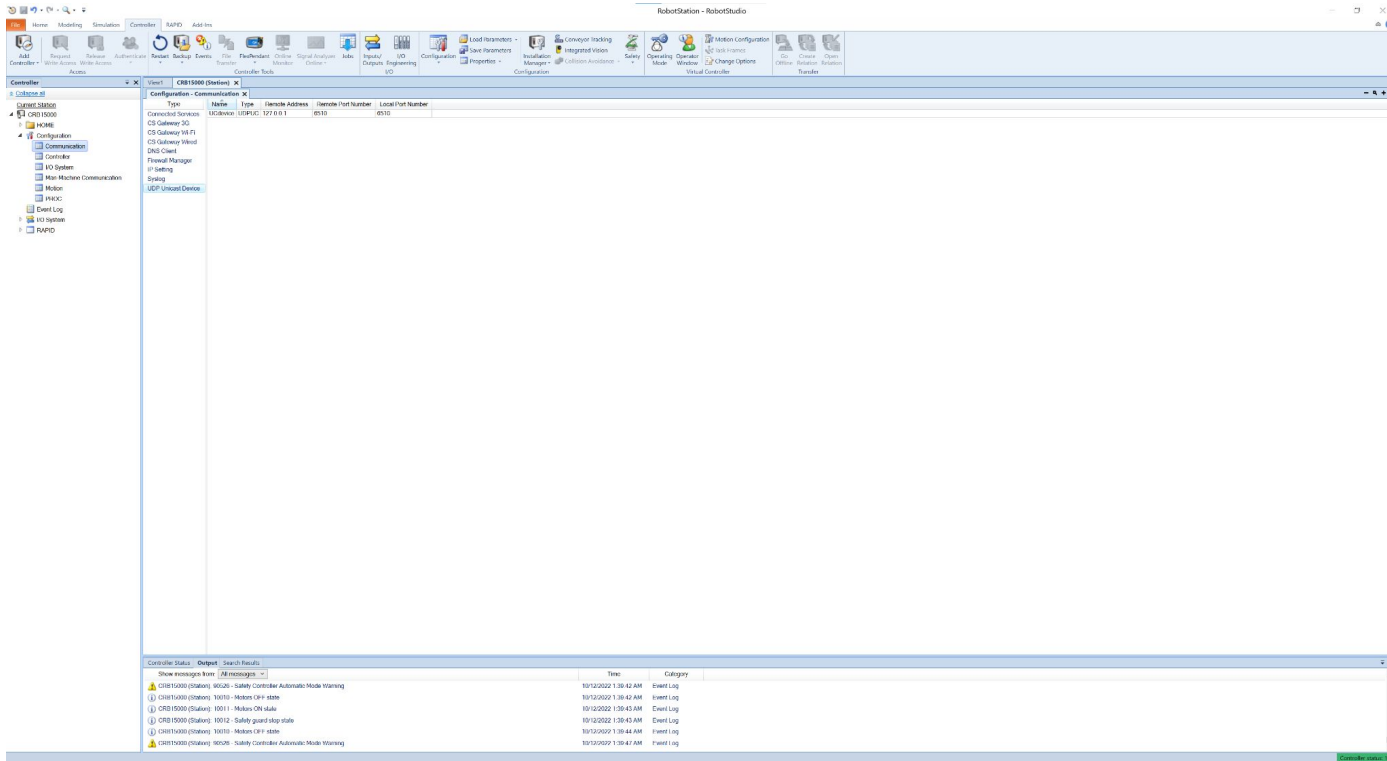
Once EGM is enabled, we need to specify which devices our controller should listen to:



Click on the **Configuration** dropdown menu on the **Controller** tab, and select the **Communication** button.

Step 2: Listening to a UDP Unicast Device

Once EGM is enabled, we need to specify which devices our controller should listen to:

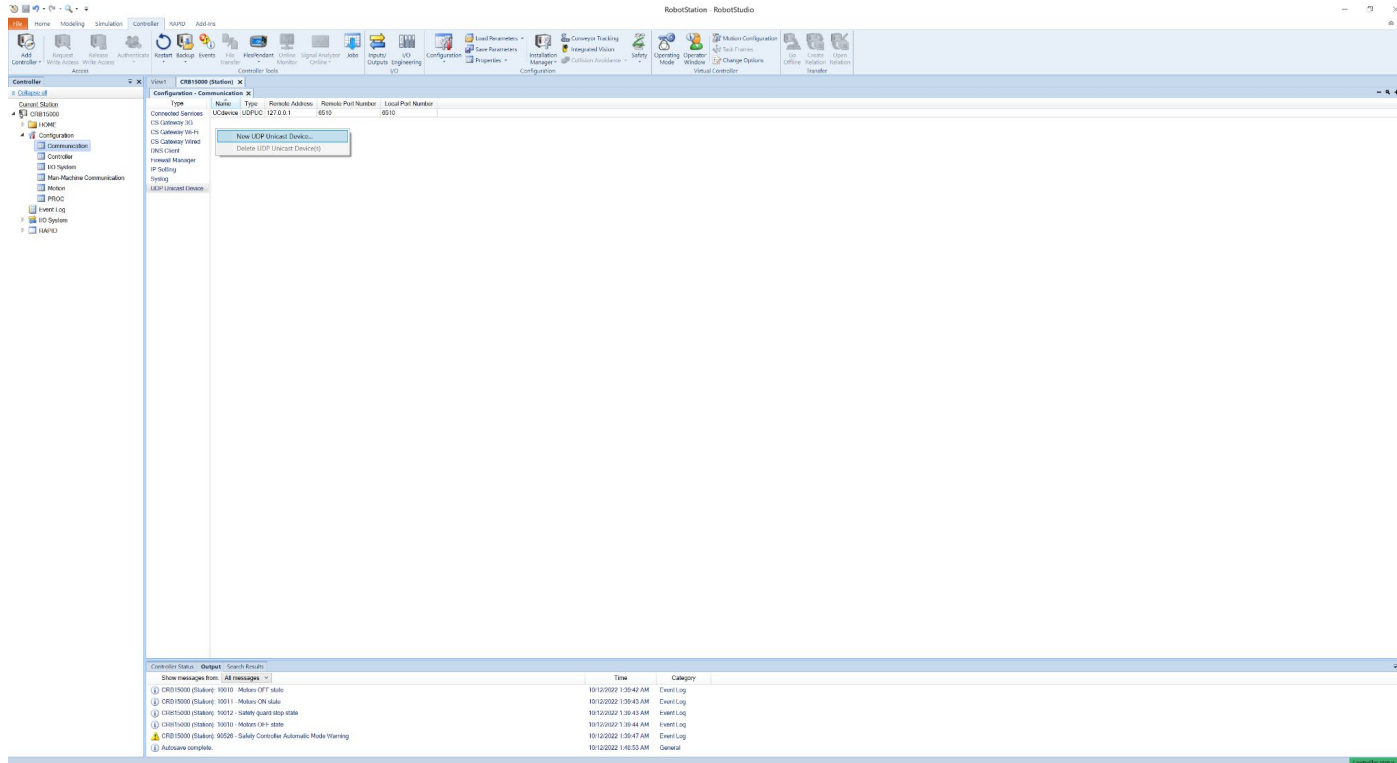


With the **Communication** window opened, select **UDP Unicast Device**.

If this option is not available for you, make sure **EGM** is available and enabled on your controller (See **Step 1**).

Step 2: Listening to a UDP Unicast Device

Once EGM is enabled, we need to specify which devices our controller should listen to:



To specify a **new external device**, you can right click on a blank space on the **UDP Unicast Device** window and select the **New UDP Unicast Device...** option.

Step 2: Listening to a UDP Unicast Device

If you are unclear on how to fill out the text fields of your new device, refer to the application manual of EGM provided by ABB:

3.3 Configuring UdpUc devices

How to configure UdpUc devices

UdpUc communicates with a maximum of eight devices over Udp. The devices act as servers, and the robot controller acts as a client. It is the robot controller that initiates the connection to the sensor.

Each UDP channel is defined as a device, i.e. you need to set up one device for each motion task where you want to use EGM.

System parameters

This is a brief description of the parameters used when configuring a device. For more information about the parameters, see *Technical reference manual - System parameters*.

These parameters belong to the type *UDP Unicast Device* in topic *Communication*.

Parameter	Description
<i>Name</i>	The name of the UDP Unicast Device instance. For example <i>EGMsensor</i> .
<i>Type</i>	The type of UDP Unicast Device protocol to be used. The only available UDP Unicast Device type is <i>UDPUC</i> .
<i>Remote Address</i>	The IP address of the external device, for example, sensor.
<i>Remote Port Number</i>	The the port number on the network node identified by <i>Remote Address</i> .
<i>Local Port Number</i>	The port number on which the controller will listen for broadcast messages.

Configuration example

The device which provides the input data for EGM, has to be configured as an UdpUc device in the following way:

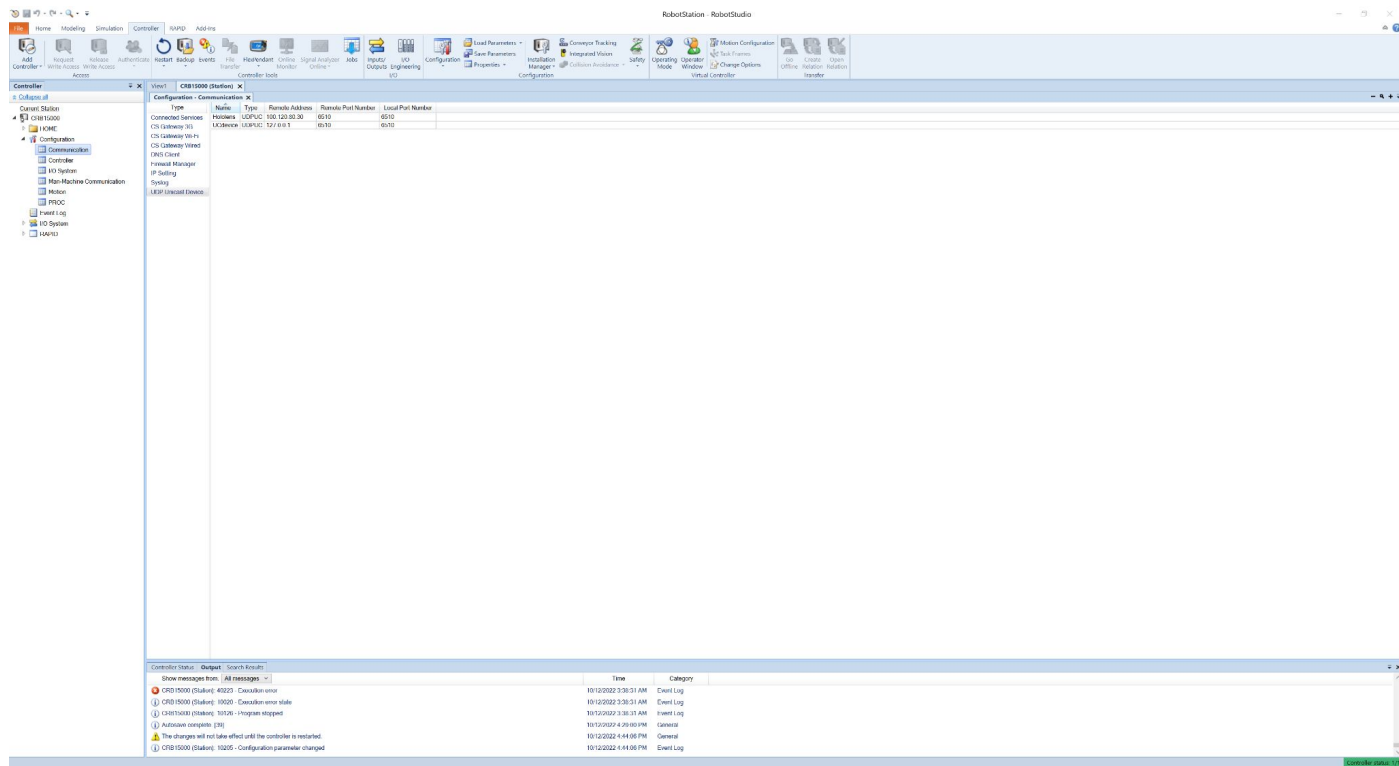
Name	Type	Remote Address	Remote Port Number
UCdevice	UDPUC	192.168.10.20	6510

Source:

<https://library.e.abb.com/public/f05090fae99a4d0ba2ee332e50865791/3HAC073318%20AM%20Externally%20Guided%20Motion%20RW7-en.pdf>

Step 2: Listening to a UDP Unicast Device

The necessary network configuration of your robot controller is done.



In my use case scenario, I am connecting an ABB robot to a Microsoft Hololens 2.

As you can see on the left image, I created a new sensor (i.e., external device) named Hololens with remote address being the IP address of my Hololens 2.

Step 3: Running EGM on RAPID

Now that your robot supports EGM communication with an external device, you need to run a RAPID code to make your robot act as an EGM client.

In the next slides I will include an example of RAPID code that you can use on your robot to guide it to a specific position (target) from an external device.

However, EGM has other functionalities that you can explore (e.g. move the robot based on axis-rotation using *EGMActJoint* and *EGMRunJoint*). Please always refer to the EGM application manual to learn how to use them.

Some other RAPID examples are also available in our repository:

<https://github.com/vcuse/egm-for-abb-robots/>

```

MODULE EGMCommunication
! Author: Felipe Franchetti
! Email: franchetti@vcu.edu
! This file is not affiliated, sponsored
! or approved by ABB. Always refer to their application
! manual for more information.

! Identifier for EGM process
VAR egmident egm_id;
! Current state of EGM process on controller
VAR egmstate egm_state;

! Convergence criteria for translation and rotation (in degrees)
CONST egm_minmax egm_minmax_translation := [-1, 1];
CONST egm_minmax egm_minmax_rotation := [-1, 1];

! Correction frame for path correction
LOCAL CONST pose egm_correction_frame := [[0, 0, 0], [1, 0, 0, 0]];
LOCAL CONST pose egm_sensor_frame := [[0, 0, 0], [1, 0, 0, 0]];

! Main function
PROC main()
    EGM_POSE_MOVEMENT;
ENDPROC

! EGM function used to move the robot to a specific position using a pose target.
! In this approach, the external device sends a message to the robot specifying angles and a position
! in the cartesian coordinate system where the robot should move to.
PROC EGM_POSE_MOVEMENT()
    ! Check if no EGM setup is active.
    IF egm_state = EGM_STATE_DISCONNECTED THEN
        TPWrite "EGM State: Preparing controller for EGM communication.";
    ENDIF

    WHILE TRUE DO
        ! Register a new EGM id.
        EGMGetId egm_id;

        ! Get current state of egm_id.
        egm_state := EGMGetState(egm_id)
        ! Setup the EGM communication.
        ! Make sure the external device name being used is the same specified in
        ! the controller communication tab. In this example, the UdpUc device name is PC,
        ! moving the robot mechanical unit ROB_1.
        IF egm_state <= EGM_STATE_CONNECTED THEN
            EGMSetupUC ROB_1, egm_id, "default", "PC", \Pose;
        ENDIF
    END

```

```

! De-serializes the message sent by the external device.
EGMActPose egm_id\Tool:=tool0,
    egm_correction_frame,
    EGM_FRAME_BASE,
    egm_sensor_frame,
    EGM_FRAME_BASE
    \x:=egm_minmax_translation
    \y:=egm_minmax_translation
    \z:=egm_minmax_translation
    \rx:=egm_minmax_rotation
    \ry:=egm_minmax_rotation
    \rz:=egm_minmax_rotation
    \LpFilter:= 16
    \MaxSpeedDeviation:=100;

! Performs a movement based on the pose target sent by the external device.
EGMRunPose egm_id, EGM_STOP_HOLD \x \y \z \rx \ry \rz \CondTime:=1\RamplTime:=0;

! (Debugging) Checks if robot is listening for external commands.
IF egm_state = EGM_STATE_CONNECTED THEN
    TPWrite "EGM State: Waiting for movement request.";
ENDIF

! (Debugging) Checks if the robot received an external command and is moving.
IF egm_state = EGM_STATE_RUNNING THEN
    TPWrite "EGM State: Movement request received. Robot is moving.";
ENDIF

! Reset EGM communication.
IF egm_state <= EGM_STATE_CONNECTED THEN
    EGMReset egm_id;
ENDIF
ENDWHILE

! (Debugging) Checks if external devices are available.
ERROR
IF ERRNO = ERR_UDPUc_COMM THEN
    TPWrite "EGM Warning: Robot is not detecting any external devices.";
    TRYNEXT;
ENDIF
ENDPROC
ENDMODULE

```

Step 4: Generating EGM.cs file for C# project


Now that our controller is ready to receive messages using EGM, it is time to prepare your EGM server:

3.2 Building an EGM sensor communication endpoint

How to build an EGM sensor communication endpoint using .Net

This guide assumes that you build and compile using Visual Studio and are familiar with its operation.

Here is a short description on how to install and create a simple test application using *protobuf-csharp-port*.

	Action
1	In Visual Studio, create a C# application.
2	Select Tools and then NuGet Package Manager , and install <i>Google.Protobuf</i> .
3	In the NuGet Package Manager , also install <i>Google.Protobuf.Tools</i> .
4	Navigate to <i>Solution package\packages\Google.Protobuf.Tools.3.xx.x\tools\windows_x64</i> .
5	Copy the EGM folder from <i>C:\ProgramData\ABB\DistributionPackages\ABB.RobotWare-7.yy\RobotPackages\RobotControl_7.zz\utility</i> to <i>\packages\Google.Protobuf.Tools.3.xx.x\tools\windows_x64</i> .
6	Open a cmd line and run " <code>.\protoc .\egm\egm.proto --csharp_out=. \egm</code> ". => Egm.cs file is built
	 Note The <i>egm.proto</i> syntax is <code>proto2</code> .
7	Add the generated file <i>egm.cs</i> to the Visual Studio project (add existing item).
8	Copy the example code into the Visual Studio Windows Console application file (<i>egm-sensor.cs</i>) and then compile, link and run.

Source:
<https://library.e.abb.com/public/f05090fae99a4d0ba2ee332e50865791/3HAC073318%20AM%20Externally%20Guided%20Motion%20RW7-en.pdf>

The generated *Egm.cs* file should be placed inside your project and used as a library. Inside the utility folder mentioned on the left image, there is a *egm-sensor.cs* file showing an example of how to use it.

In our repository, we also make available different projects using *Egm.cs* (from WPF to Unity applications).

Please visit:

<https://github.com/vcuse/egm-for-abb-robots/>