



**UNCUYO**  
UNIVERSIDAD  
NACIONAL DE CUYO



**FACULTAD  
DE INGENIERÍA**



## ***Simulación Cinemática de Robot de 6 GDL ABB IRB 1410 en Realidad Aumentada***

*Realidad Virtual  
Ingeniería Mecatrónica  
2022*

*Reinoso, Maximiliano Gabriel - L:11754  
Campanella, Franco Emanuel - L: 12125*

# Índice

<b>Resumen</b>	<b>3</b>
<b>Introducción</b>	<b>4</b>
<b>Desarrollo</b>	<b>5</b>
Entornos utilizados	5
Unity	5
Blender	5
Visual Studio Code	5
Vuforia Engine	6
Arduino IDE	6
Implementación en Unity (Software)	7
Imagen de referencia (Target)	8
Operaciones realizadas durante el desarrollo	9
Tratamiento del modelo 3D - "BLENDER"	9
Vinculación del modelo 3D con una imagen Target - "VUFORIA"	11
Desarrollo de la aplicación android - "UNITY"	11
Script desarrollado en Visual Studio Code / C#	14
Resultado obtenido (Aplicación Android)	18
Implementación en Arduino de un control Bluetooth (Hardware)	19
Código C++ en Arduino IDE	20
Lista de materiales utilizados en el Joystick	21
Resultado obtenido (Joystick)	21
<b>Librerías y recursos extras:</b>	<b>22</b>
<b>Futuras mejoras</b>	<b>22</b>
<b>Conclusión</b>	<b>23</b>
<b>Bibliografía</b>	<b>23</b>

## Resumen

*Este proyecto se desarrolla para la materia de Realidad Virtual de la carrera Ingeniería en Mecatrónica de la Facultad de Ingeniería de la Universidad Nacional de Cuyo con el objetivo de realizar la simulación, mediante Realidad Aumentada, de la cinemática de un Robot de 6 GDL modelo ABB IRB 1410.*

*Este desarrollo tiene como finalidad visualizar a dicho Robot en las condiciones de entorno y trabajo que tendría el dispositivo físico real en una determinada aplicación. La idea de esta aplicación es permitir realizar una visualización del mismo, durante el proceso de selección y, de esta forma, validar los pasos siguientes del proceso de adquisición, instalación y puesta en marcha de la unidad real, abaratando los costos administrativos y sobre todo técnicos que llevaría realizar el ciclo del proceso de selección por personal de la planta.*

## Introducción

Debido a la gran cantidad de industrias en desarrollo en los últimos tiempos, la incorporación de sistemas autónomos, en especial brazos robóticos, ha crecido notablemente. A su vez, existen varias empresas que desarrollan estos sistemas, las cuales, poseen una gran variedad de opciones para satisfacer las necesidades del cliente. Por lo tanto, a la hora de realizar la tarea de selección del mejor candidato para nuestra aplicación específica nos encontramos con que hay varias opciones que podemos utilizar y esto trae consigo una serie de interrogantes:

- **¿Disponemos del espacio físico necesario para la instalación de determinada unidad?**
- **¿Debemos realizar modificaciones físicas en la planta para la instalación de la unidad?**
- **¿Cómo vamos a hacer llegar la materia prima al robot y cómo será retirada luego de ser manipulada?**
- **¿Cumple con las necesidades dimensionales que necesitamos?**
- **¿Qué otros elementos necesitamos tener o cambiar para que funcione adecuadamente?**

Todas estas preguntas recaen en tener que dedicar horas de trabajo de personal técnico para la toma de decisiones basándonos en las dimensiones, restricciones, necesidades energéticas y de seguridad que ofrece o exige el fabricante. Esto obliga a disponer de información que se debe relevar de la instalación de la planta. Una vez que se dispone de toda esta información, se puede comenzar con la tarea de selección, la cual obliga a revisar la información adquirida o recolectar datos faltantes en caso de ser necesario.

Todo este trabajo se podría agilizar notablemente si se dispusiera de un robot real a escala a partir del cual se diseñe toda la instalación. Pero debido a los altos costos que esto supone, se hace difícil de realizar.

Esta propuesta trae al cliente una herramienta que permite tener este modelo de dimensiones reales mediante el uso de Realidad Aumentada. Esto disminuiría notablemente los tiempos de dimensionamiento del espacio y la instalación de la unidad y de los elementos anexos que este necesite.

## Desarrollo

*Para el desarrollo de esta aplicación se utilizaron algunos softwares conocidos.*

### Entornos utilizados

#### Unity

*El software empleado para realizar el entorno virtual es Unity. Éste es un motor de videojuego multiplataforma creado por Unity Technologies. Está disponible como plataforma de desarrollo para Microsoft Windows, OS X, Linux y Android, entre otros. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas.*



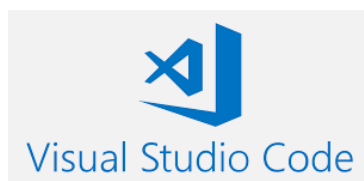
#### Blender

*En dicho software es posible realizar tanto aplicaciones 2D como 3D. Soporta varios formatos de archivos tales como 3ds Max, Maya, Softimage, Blender, Modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, entre otros. Dispone además de herramientas y librerías que permiten con facilidad realizar interfaces IU. También dispone de un motor de física, para agregar efectos realistas a los cuerpos. También cuenta con una tienda Asset Store, donde se ofrecen miles de recursos, herramientas y extensiones, del cual ha sido tomada la base del entorno actual.*



#### Visual Studio Code

*Unity además soporta el lenguaje de programación C#, el cual será empleado durante el desarrollo, y su edición se realizará a través del entorno de desarrollo Visual Studio Code. Como dato importante, se aclara que el compilador empleado es el de Unity y no el de Visual Studio Code, lo cual traerá posteriormente problemas de compatibilidad de librerías.*



## Vuforia Engine

*Vuforia Engine es utilizado para crear aplicaciones de realidad aumentada para Android, iOS y UWP para dispositivos móviles y anteojos AR. Las aplicaciones se pueden crear con Unity, Android Studio, Xcode y Visual Studio. Vuforia Engine se puede importar fácilmente a Unity y, por lo tanto, es una herramienta muy utilizada en la creación de juegos y simuladores.*



## Arduino IDE

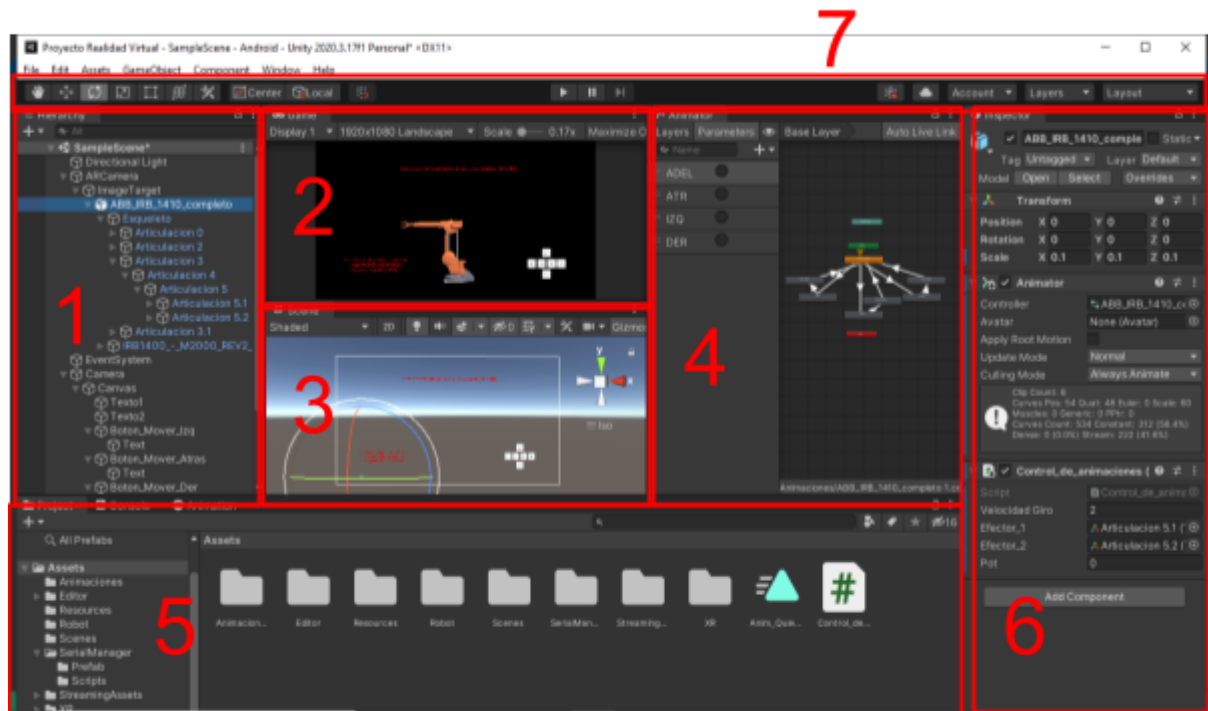
*El IDE Arduino es un software de código abierto basado en Processing que permite fácilmente escribir código y cargar el programa ya compilado a la memoria flash de la placa Arduino por el puerto serie. Además, Arduino es una plataforma mundialmente difundida para la cual existen gran cantidad de módulos y librerías que permiten adaptar e implementar en cualquier plataforma. Unity no es la excepción a esto, posee librerías tanto para comunicación serie como para comunicación Bluetooth, entre otros. También permite interactuar con la electrónica asociada por lo que lo hace una herramienta muy útil para la implementación de IHM.*



*Finalmente, se puede observar que Unity es un programa sumamente versátil, poderoso y relativamente fácil de utilizar. Además, éste software en la actualidad tiene abundante documentación para poder comprender y utilizar al máximo sus prestaciones. Es por esto que el desarrollo de la aplicación se basó en la utilización de la documentación presente en la web, además de los distintos foros referidos al proyecto en cuestión.*

## Implementación en Unity (Software)

A continuación, se describen algunos de los elementos el entorno de unity:



**1) Hierarchy Window:** es una representación de la jerarquía de cada objeto en la escena. Cada elemento en Scene View tiene una entrada en la jerarquía, por lo que las dos ventanas están inherentemente vinculadas. La jerarquía revela la estructura de cómo los objetos están agrupados el uno al otro.

**2) Game:** permite visualizar la representación obtenida del juego o escena desarrollada a fin de visualizar cómo se vería el resultado por el usuario. También permite cuando se está corriendo la aplicación interactuar con la escena a fin de poder probar los elementos añadidos como botones, sliders, entre otros elementos.

**3) Scene View:** permite una navegación visual y editar la escena del juego. Puede mostrar una perspectiva 2D o 3D dependiendo del tipo de proyecto en el que se esté trabajando. Aquí se ubican todos los objetos que conforman la escena, en donde se pueden manipular y editar.

**4) Animator Controller:** permite relacionar las animaciones que incorpora nuestra aplicación, el pase de un estado a otro y programar las acciones que desencadenan el flujo de las animaciones dentro de la aplicación. Esto permite desarrollar secuencias de animaciones o asignar animaciones a los elementos o personajes, por ejemplo, correr o saltar.

**5) Project:** muestra los assets de librería que están disponibles para ser usados. Un asset es una representación de cualquier ítem que puede ser utilizado en un



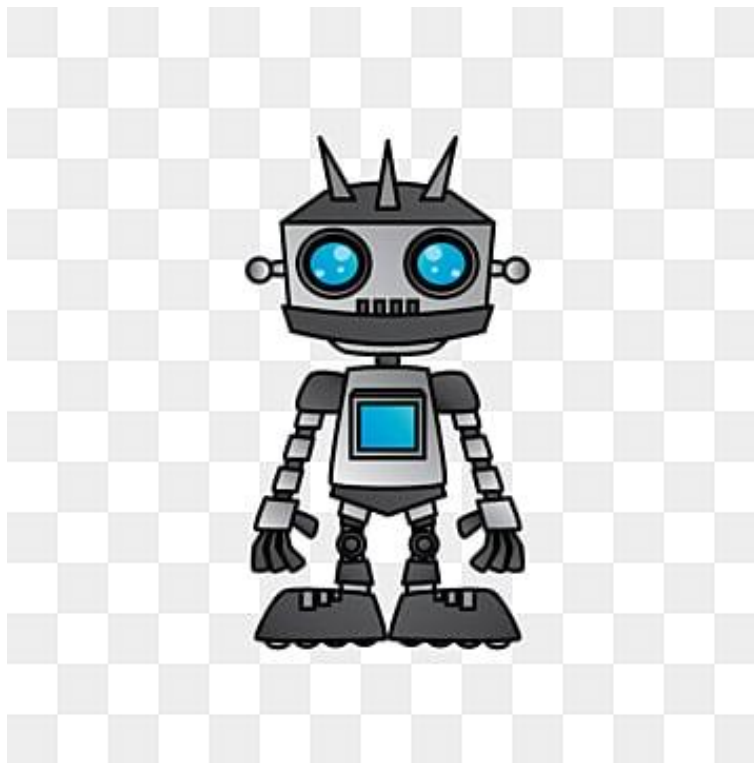
juego o proyecto. Un asset podría venir de un archivo creado fuera de Unity, tal como un modelo 3D, un archivo de audio, una imagen, o cualquiera de los otros tipos de archivos que Unity soporta. También hay otros tipos de asset que pueden ser creados dentro de Unity, tal como un Animator Controller, un Audio Mixer o una Render Texture.

**6) Inspector:** permite visualizar y editar todas las propiedades del objeto seleccionado. Ya que diferentes objetos tienen diferentes propiedades, el diseño y contenido de la ventana del inspector va a variar según sea el GameObject seleccionado.

**7) Barra de Herramientas:** proporciona un acceso a las características más esenciales para trabajar. En la izquierda contiene las herramientas básicas para manipular la Scene View y los objetos dentro de esta. En el centro están los controles de reproducción, pausa, y pasos. Los botones a la derecha dan acceso a los servicios de Unity Cloud y cuenta de Unity, seguido por un menú de visibilidad de capas, y finalmente el menú del layout del editor (que proporciona algunos diseños alternativos para la ventana del editor).

Imagen de referencia (*Target*)

La siguiente imagen es utilizada como “**TARGET**” por la aplicación, de modo que cuando se detecta por la cámara, posiciona el modelo 3D en Realidad Aumentada, permitiendo ejecutar las acciones de simulación.



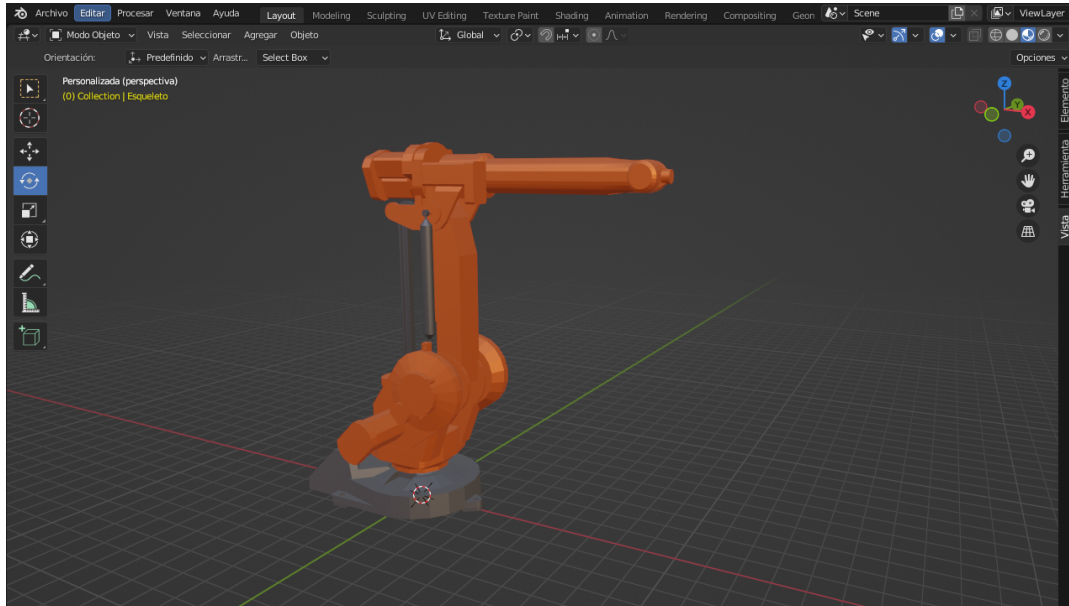


## Operaciones realizadas durante el desarrollo

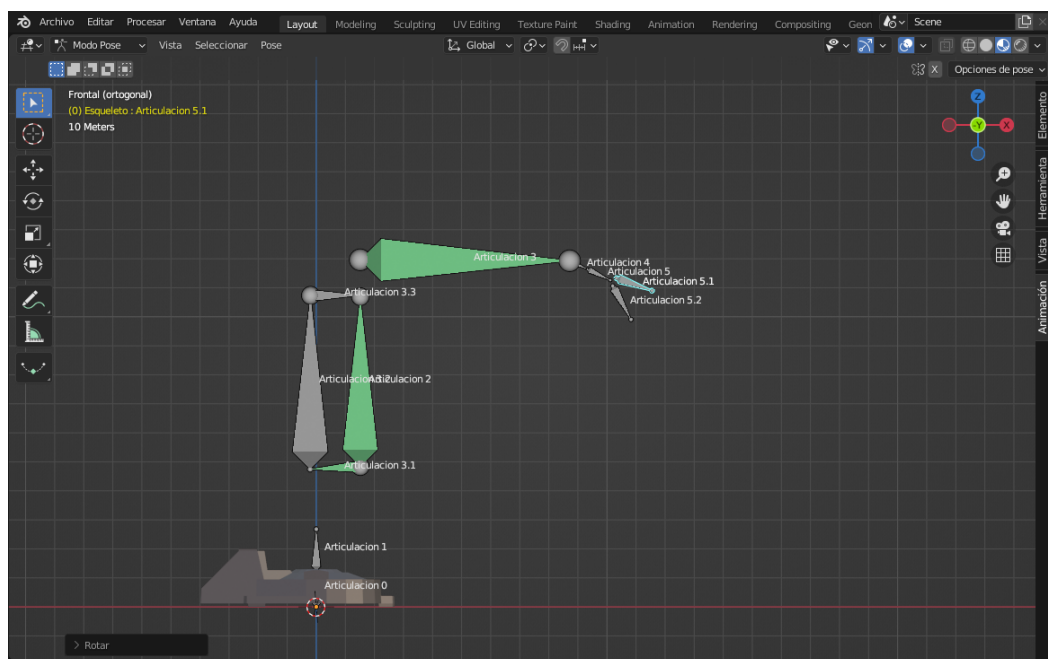
### 1. Tratamiento del modelo 3D - “BLENDER”

*En primera instancia, se trabajó con un modelo 3D del robot comercial ABB IRB 1410 ofrecido por el fabricante. Se consiguió un único archivo “.stl” que contenía el modelo del robot a escala real y representado en un único objeto 3D.*

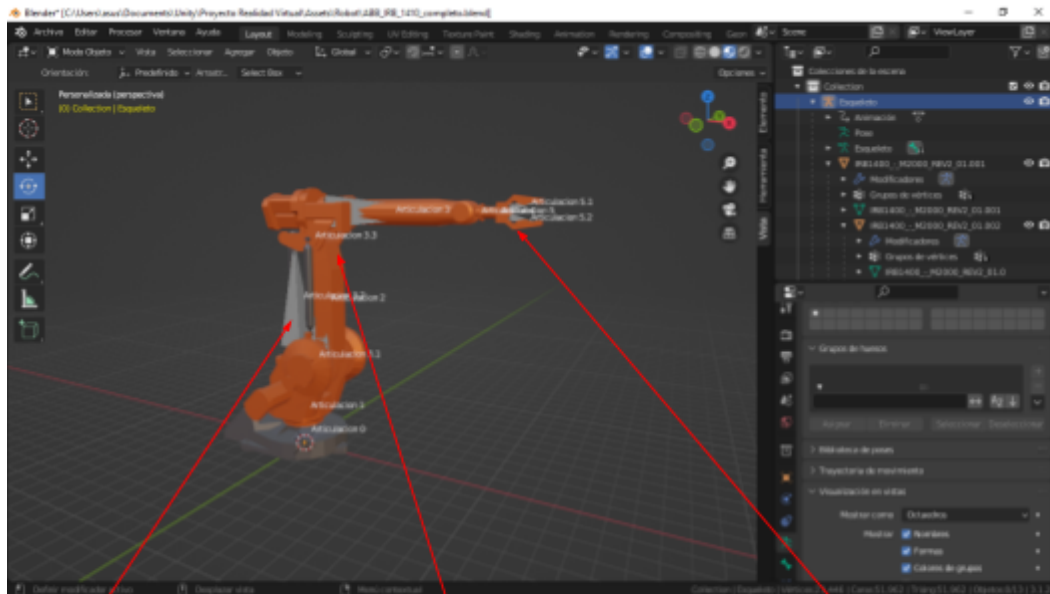
*Posteriormente, se separó cada pieza del cuerpo en objetos independientes.*



*Para la vinculación rotacional y longitudinal de las piezas, se incorporaron elementos llamados “huesos” del objeto, los cuales cumplen la función de articular el modelo 3D del robot.*



Luego, se agregó un efector final que fuera útil y representativo para la aplicación en cuestión, del tipo pinza, al cual se le realizó un tratamiento similar al descrito anteriormente.



Huesos del modelo

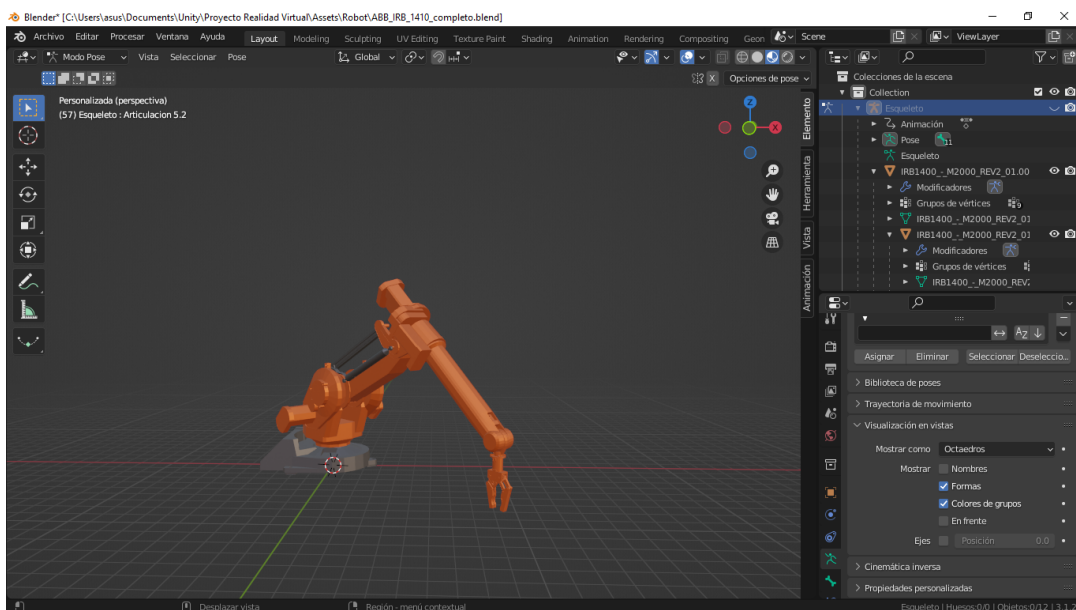
Cuerpo completo separado en articulaciones

Efector final

Por último, se diseñaron 4 animaciones típicas de movimiento del robot que representan operaciones de "pick and place" para el mismo:

- Movimiento de Pick Up hacia adelante.
- Movimiento de Pick Up a la derecha.
- Movimiento de Pick Up a la izquierda.
- Movimiento de homing o retorno a la posición de origen.

A continuación, se muestra una de las posiciones características de esta operación:



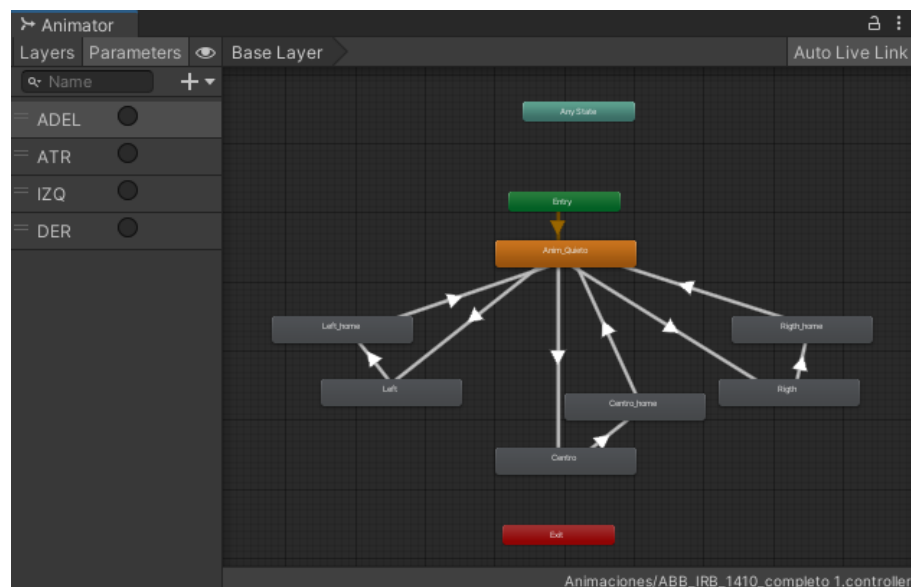
## 2. Vinculación del modelo 3D con una imagen Target - “VUFORIA”

Una vez finalizado el tratamiento del modelo 3D en Blender, se incorpora a Unity, donde será referenciado mediante una imagen Target. Dicha imagen fue previamente cargada a la página de Vuforia, donde se analizó su rendimiento como referencia, y luego, se descargó para poder ser importada al proyecto en Unity. También, se debe instalar la herramienta Vuforia como complemento en Unity lo cual se hace desde la página descargando un archivo del tipo “UnityPackage”.

Luego de la instalación, en la jerarquía de Unity, se debe agregar un elemento “ARCamera” y, dentro de este, un elemento hijo “ImageTarget”. Dentro de este elemento incorporamos el modelo 3D resultante de Blender y, de esta forma, cuando se detecte la imagen target por la aplicación, se desplegará el modelo 3D correspondiente.

## 3. Desarrollo de la aplicación android - “UNITY”

Una vez referenciado el modelo 3D al target Vuforia, se procedió a diseñar una lógica de control de las 4 animaciones elaboradas en Blender. Para esto se hizo uso de la herramienta “Animator Controller” dispuesta por Unity. Este objeto de software permite ejecutar alguna de las animaciones de movimiento desarrolladas en Blender mediante alguna acción de entrada comandada por el usuario. Lo anterior se resuelve por medio de variables del tipo “Trigger” que se activan con algún comando de entrada y se desactivan al ejecutar la acción programada. De este modo, es que se realiza la vinculación de un botón, físico o virtual, con la ejecución de una acción por software.



En la imagen anterior se ve representado el “Animator controller” de la aplicación. Este incluye cada una de las animaciones de avance con sus respectivos movimientos de homing. Cada uno de los casilleros representa un estado del robot, y cada una de las flechas, la acción de activación que ejecuta una transición entre dos estados.

Del lado izquierdo de la pantalla quedan representadas las variables del tipo “Trigger”, cuya activación se realiza desde el script desarrollado en C#.

Luego de todo este desarrollo sólo resta por incorporar a la pantalla los elementos de IHM, botones y slider. Estos se vinculan incorporándose como objetos hijos del objeto pantalla (“Canvas”). Son piezas del tipo “UI” (User Interface) que son agregadas y editadas a conformidad. Dichos elementos están vinculados a una porción de código que representa su comportamiento. A continuación se muestran los scripts correspondientes:

- *Lógica de slider para escalar robot:*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class SliderEscalar : MonoBehaviour
{
    public GameObject modelos3D;
    public Slider sliderEscalar;

    public void EscalarModelo()
    {
        modelos3D.transform.localScale = new Vector3(sliderEscalar.value, sliderEscalar.value, sliderEscalar.value);
    }
}
```

- *Lógica de botones izquierda, derecha, adelante y atrás respectivamente:*

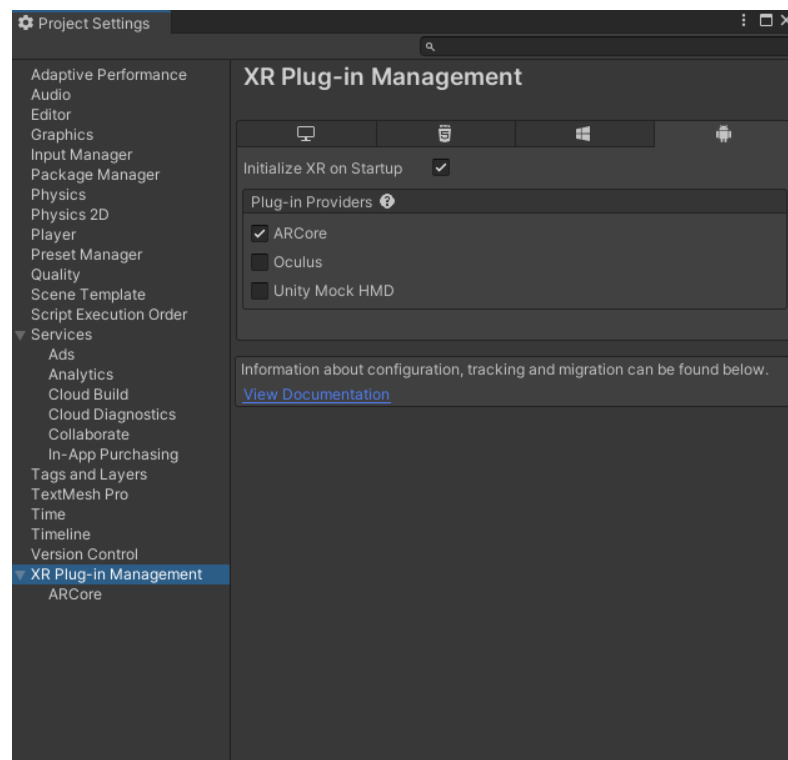
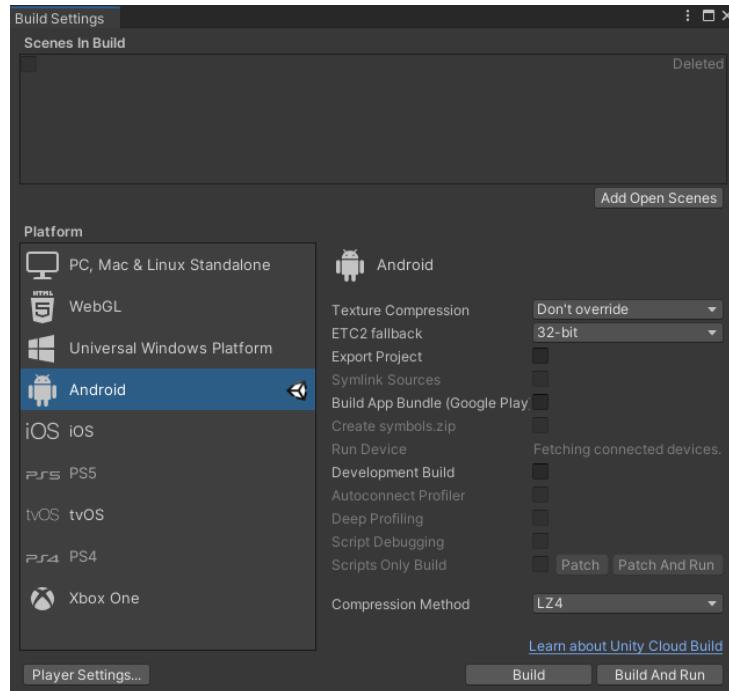
```
//Control por pantalla
public void Mover_Izq(){
    anim.SetTrigger("IZQ");
}

public void Mover_Atras(){
    anim.SetTrigger("ATR");
}

public void Mover_Der(){
    anim.SetTrigger("DER");
}

public void Mover_Adel(){
    anim.SetTrigger("ADEL");
}
```

Luego, una vez logrado el resultado de la IHM y de la representación y manipulación del modelo 3D, se procede a exportar la aplicación para plataforma Android. Esto se realiza desde “File”, “Build Settings”. Aquí, se debe seleccionar la plataforma Android, para que la aplicación funcione correctamente se deben configurar algunas variables desde el botón “Player Settings”. Además, se debe instalar el complemento XR Plug-in Management, el cual permite correr aplicaciones en Realidad Aumentada.



Finalmente, se procede a generar la aplicación mediante el botón “Build” y se obtiene como resultado la aplicación desarrollada.



Script desarrollado en Visual Studio Code / C#

*A continuación, se muestra el código utilizado para desarrollar la Comunicación Bluetooth y el Control de los Movimientos del Robot. En dicho código se muestran las siguientes funciones:*

```
void Start()
```

*Inicializa variables y crea el Objeto Bluetooth, el cual llevará adelante la comunicación Bluetooth.*

```
private void ReceiveData(string dato)
```

*Esta función recibe el dato desde el Joystick (Arduino UNO) mediante comunicación Bluetooth (HC-05) e interpreta el mismo. Como resultado de la interpretación, surge la ejecución de diferentes acciones:*

- **Mover Robot hacia Adelante, mediante la función Mover\_Adel()**
- **Mover Robot hacia Atrás, mediante la función Mover\_Atras()**
- **Mover Robot hacia Izquierda, mediante la función Mover\_Izq()**
- **Mover Robot hacia Derecha, mediante la función Mover\_Der()**
- **Abrir o Cerrar el Efecto Final, mediante la función RotateEfecto()**

```
void RotateEfecto(bool efecto_cerrar, bool efecto_abrir)
```

*Esta función recibe como parámetro dos valores booleanos (efecto\_abrir, bool efecto\_cerrar), los cuales representan dentro de la función la acción que se debe llevar a cabo, es decir, abrir o cerrar el efecto final. Dependiendo del valor lógico de dichas variables, se decidirá entre ejecutar las funciones Abrir\_Efecto() o Cerrar\_Efecto(), respectivamente.*

```
void Update()
```

*Esta función comprueba si se ha establecido la Comunicación Bluetooth, en caso de estar conectado con un dispositivo, realiza la recepción del dato y lo envía a la función ReceiveData() para ser interpretado. Además, esta función incorpora la lectura de la activación de ciertas letras del teclado. Esto, en un principio, era una funcionalidad para la aplicación desarrollada para PC, pero se decidió mantenerla en caso de que el usuario lo requiera.*

```
StartButton() y StopButton()
```

*Estas funciones establecen la conexión y desconexión de la aplicación con el Joystick desarrollado, respectivamente. El dispositivo Bluetooth debe tener el nombre “RV” para que se establezca la conexión. La función StopButton() además, realiza el cierre de la aplicación.*



```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Control_de_animaciones : MonoBehaviour
{
    //Control BT
    public Text deviceName;
    private bool IsConnected;
    public static string dataReceived = "";

    //Control de movimiento
    public float velocidadGiro;
    public Transform Efecto_1;
    public Transform Efecto_2;
    Animator anim;

    //Control por HW
    public int pot, boton;
    string dato_ant = "5000";
    int pot_ant = 0;
    bool efector_cerrar = false, efector_abrir = false;
    string efector;

    // Start is called before the first frame update
    void Start()
    {
        //Inicializacion BT
        IsConnected = false;
        BluetoothService.CreateBluetoothObject();

        //Control brazo
        anim = GetComponent<Animator>();
    }

    //Control por bluetooth
    private void ReceiveData(string dato){
        if(dato_ant != dato){
            int.TryParse(dato, out boton);
            if(boton == 3000){
                anim.SetTrigger("IZQ");
            }else if(boton == 2000){
                anim.SetTrigger("DER");
            }else if(boton == 4000){
                anim.SetTrigger("ATR");
            }else if(boton == 5000){
                anim.SetTrigger("ADEL");
            }else{
                int.TryParse(dato, out pot);
                if(pot > pot_ant){
                    efector_abrir = false;
                    efector_cerrar = true;
                    efector = "cerrar";
                }else if ((pot < pot_ant)){
                    efector_cerrar = false;
                    efector_abrir = true;
                    efector = "abrir";
                }
                //Debug.Log(efector);
                Debug.Log(dato);
                RotateEfector(efector_cerrar, efector_abrir);
                pot_ant = pot;
            }
            dato_ant = dato;
        }
    }
}
```



```
void RotateEfector(bool efector_cerrar, bool efector_abrir)
{
    float z1 = Efector_1.localRotation.eulerAngles.z;
    float z2 = Efector_2.localRotation.eulerAngles.z;
    if(z2 > 2){
        if(efector_cerrar){
            Efector_2.Rotate(new Vector3(0f, 0f, -velocidadGiro));
            Efector_1.Rotate(new Vector3(0f, 0f, velocidadGiro));
            efector_cerrar = false;
        }
    }
    if(z2 < 50){
        if(efector_abrir){
            Efector_2.Rotate(new Vector3(0f, 0f, velocidadGiro));
            Efector_1.Rotate(new Vector3(0f, 0f, -velocidadGiro));
            efector_abrir = false;
        }
    }
}

//Control por teclado de PC
void Update()
{
    if (IsConnected) {
        try
        {
            string datain = BluetoothService.ReadFromBluetooth();
            if (datain.Length > 0)
            {
                dataRecived = datain;
                ReceiveData(dataRecived);
            }
        }
        catch (Exception e)
        {
        }
    }

    if(Input.GetKeyDown(KeyCode.A)){
        anim.SetTrigger("IZQ");
    }

    if(Input.GetKeyDown(KeyCode.S)){
        anim.SetTrigger("ATR");
    }

    if(Input.GetKeyDown(KeyCode.D)){
        anim.SetTrigger("DER");
    }

    if(Input.GetKeyDown(KeyCode.W)){
        anim.SetTrigger("ADEL");
    }

    if(Input.GetKey(KeyCode.E))
    {
        efector_abrir = false;
        efector_cerrar = true;
        RotateEfector(efector_cerrar, efector_abrir);
    }
    if(Input.GetKey(KeyCode.Q)){
        efector_abrir = true;
        efector_cerrar = false;
        RotateEfector(efector_cerrar, efector_abrir);
    }
}
```

```
//Boton Conectar
public void StartButton()
{
    if(!IsConnected){
        IsConnected = BluetoothService.StartBluetoothConnection("RV\r");
    }
    if(!IsConnected){
        IsConnected = BluetoothService.StartBluetoothConnection("RV");
    }
}

//Boton Desconectar
public void StopButton()
{
    if (IsConnected)
    {
        BluetoothService.StopBluetoothConnection();
    }
    Application.Quit();
}

//Control por pantalla
public void Mover_Izq(){
    anim.SetTrigger("IZQ");
}

public void Mover_Atras(){
    anim.SetTrigger("ATR");
}

public void Mover_Der(){
    anim.SetTrigger("DER");
}

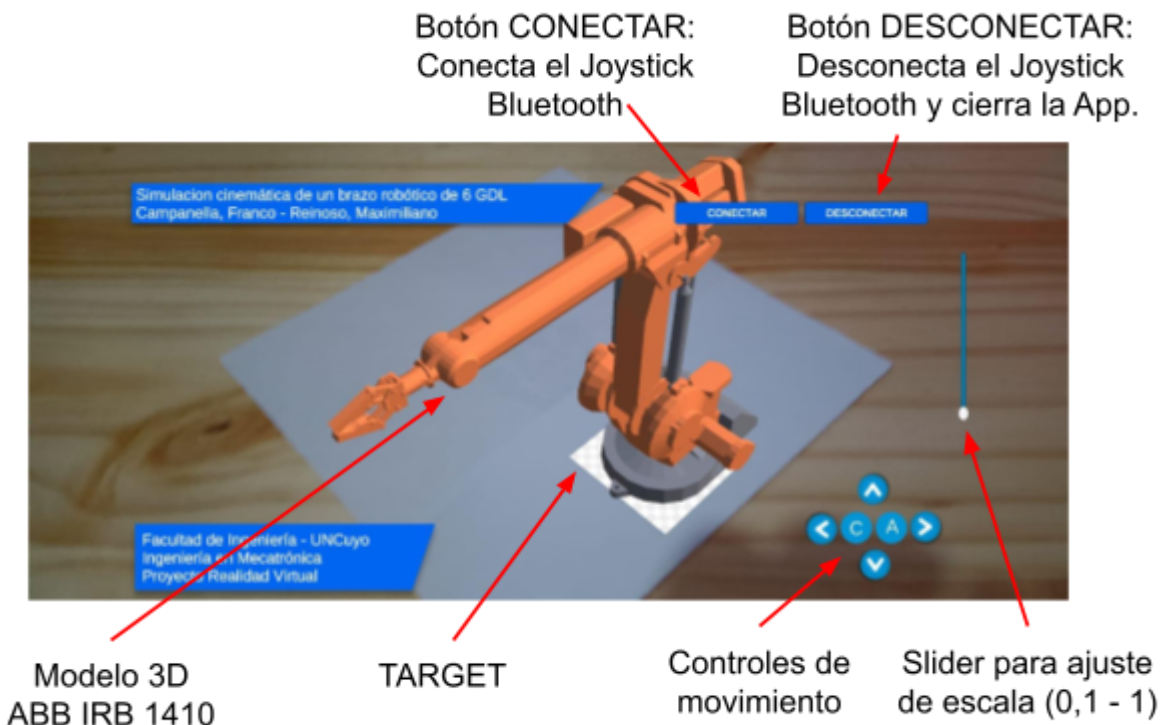
public void Mover_Adel(){
    anim.SetTrigger("ADEL");
}

public void Cerrar_Efector(){
    efector_abrir = false;
    efector_cerrar = true;
    RotateEfector(efector_cerrar, efector_abrir);
}

public void Abrir_Efector(){
    efector_abrir = true;
    efector_cerrar = false;
    RotateEfector(efector_cerrar, efector_abrir);
}
}
```

## Resultado obtenido (Aplicación Android)

En la siguiente imagen se puede ver una captura de pantalla del resultado del desarrollo de la aplicación.

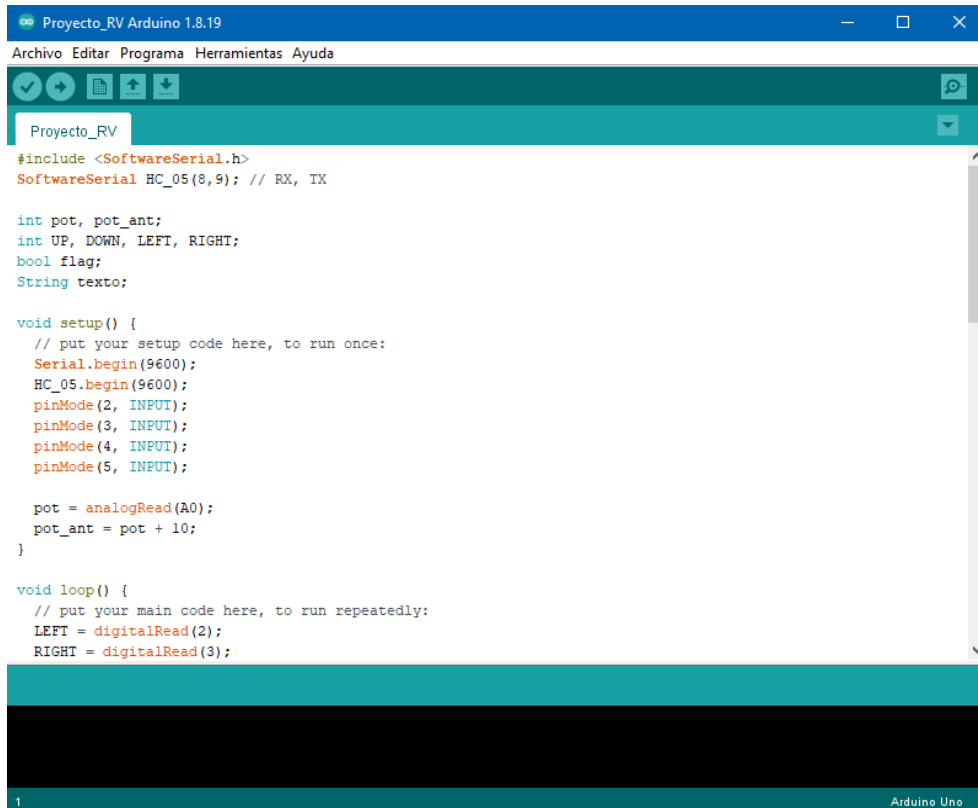


En la pantalla se pueden ver los siguientes elementos:

- **BOTÓN CONECTAR:** Al ser presionado, establece la comunicación Bluetooth con el Joystick permitiendo la recepción de comandos de entrada.
- **BOTÓN DESCONECTAR:** Al ser presionado, finaliza la comunicación Bluetooth con el Joystick y luego, cierra la aplicación.
- **MODELO 3D:** Corresponde al modelo 3D utilizado en el desarrollo de la aplicación, dicho modelo será animado mediante los comandos recibidos por Bluetooth o ingresados por los controles en pantalla por el usuario.
- **TARGET:** Imagen utilizada como referencia de tamaño y posición para referenciar el modelo 3D utilizado.
- **CONTROLES DE MOVIMIENTO:** Permiten ingresar los comandos de mover adelante, atrás, izquierda, derecha, abrir y cerrar el efector final mediante pantalla y de esta forma poder independizarlo del Joystick o disponer de ambas funcionalidades si se requiere.
- **SLIDER PARA AJUSTE DE ESCALA:** Permite ajustar la escala del modelo 3D dentro de un rango que va desde el 10% al 100% del tamaño real.

## *Implementación en Arduino de un control Bluetooth (Hardware)*

*A continuación, se muestra una imagen del entorno de desarrollo implementado para configurar el hardware: Arduino IDE*



```
Proyecto_RV Arduino 1.8.19
Archivo Editar Programa Herramientas Ayuda

Proyecto_RV

#include <SoftwareSerial.h>
SoftwareSerial HC_05(8,9); // RX, TX

int pot, pot_ant;
int UP, DOWN, LEFT, RIGHT;
bool flag;
String texto;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  HC_05.begin(9600);
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);

  pot = analogRead(A0);
  pot_ant = pot + 10;
}

void loop() {
  // put your main code here, to run repeatedly:
  LEFT = digitalRead(2);
  RIGHT = digitalRead(3);
}
```



## Código C++ en Arduino IDE

```
#include <SoftwareSerial.h>
SoftwareSerial HC_05(8,9); // RX, TX

int pot, pot_ant;
int UP, DOWN, LEFT, RIGHT;
bool flag;
String texto;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  HC_05.begin(9600);
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);

  pot = analogRead(A0);
  pot_ant = pot + 10;
}

void loop() {
  // put your main code here, to run repeatedly:
  LEFT = digitalRead(2);
  RIGHT = digitalRead(3);
  UP = digitalRead(4);
  DOWN = digitalRead(5);
  pot = analogRead(A0);

  flag = true;

  if(!UP){
    delay(150);
    Serial.println("w");
    HC_05.println((String) 5000 + "\r");
    //HC_05.println("w");
    flag = false;
  }
  if(!DOWN){
    delay(150);
    Serial.println("s");
    HC_05.println((String) 4000 + "\r");
    flag = false;
  }
  if(!LEFT){
    delay(150);
    Serial.println("a");
    HC_05.println((String) 3000 + "\r");
    flag = false;
  }
  if(!RIGHT){
    delay(150);
    Serial.println("d");
    HC_05.println((String) 2000 + "\r");
    flag = false;
  }

  if(Serial.available())
  {
    texto = Serial.readString();
    HC_05.println(texto); //Forward what Serial received to Software Serial Port
  }
  if(HC_05.available())
  {
    Serial.write(HC_05.read()); //Forward what Software Serial received to Serial Port
  }

  if(flag)delay(100);

  if((pot > (pot_ant+4)) || (pot < (pot_ant-4))){
    Serial.println((String) pot + "\r");
    HC_05.println((String) pot + "\r");
    pot_ant = pot;
  }
}
```

*El código anterior se implementó en una placa Arduino UNO (ATMega328p), microcontrolador que comanda el Hardware (Joystick). Este, recibe las señales eléctricas de entrada provenientes de los pulsadores y el potenciómetro, y las comunica mediante Bluetooth (módulo HC-05) al dispositivo Android en el cual se está ejecutando la aplicación.*

*Dentro de las partes más importantes de dicho código se tiene:*

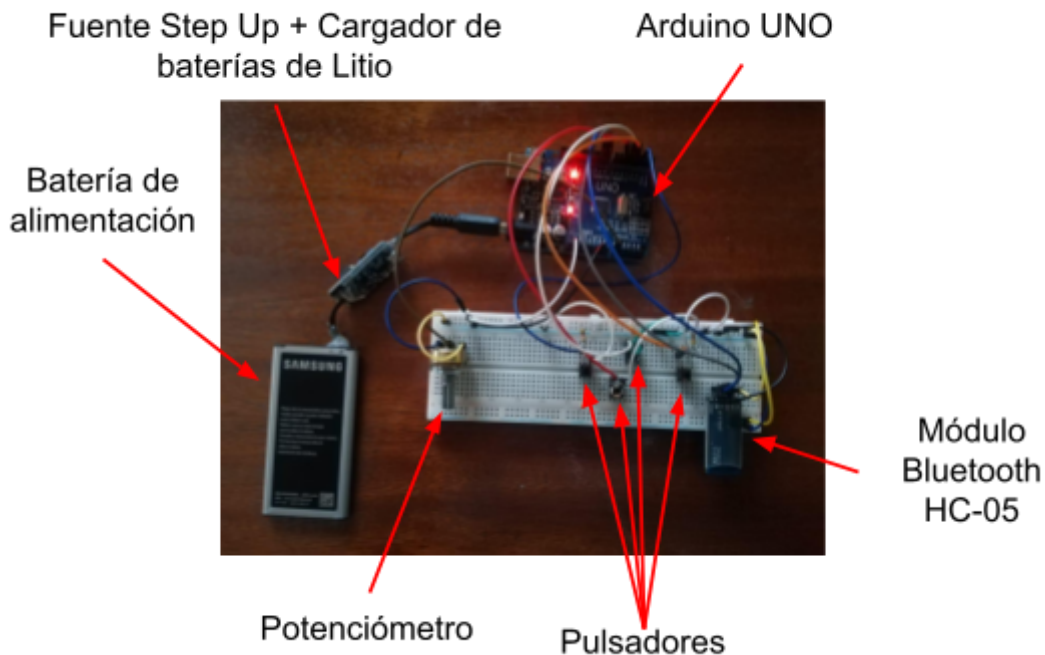
- En primer lugar, la utilización de condicionales, en los cuales dependiendo del pulsador accionado es el comando que será transmitido por Bluetooth.*
- En segundo lugar, un último condicional el cual compara el valor del potenciómetro con su valor anterior y en caso de encontrar una diferencia que no sea producto del ruido de la señal de entrada, envía el valor a través del Bluetooth.*

#### **Lista de materiales utilizados en el Joystick**

- *Potenciómetro de 10k $\Omega$  (1 un.).*
- *Pulsadores Tact Switch (4 un.).*
- *Placa Arduino UNO (1 un.).*
- *Módulo Bluetooth HC-05 (1 un.).*
- *Resistencias de 4,7k $\Omega$  (4 un.).*
- *Batería de celular de 3.8V / 2800mAh (1 un.).*
- *Fuente Step Up de tensión MT 3608 (1 un.).*
- *Modulo Cargador De Baterías de Litio Tp4056 (1 un.).*
- *Conector Plug de alimentación (1 un.).*
- *Cables para conexión de los elementos.*

#### **Resultado obtenido (Joystick)**

*La siguiente imagen indica los componentes de hardware utilizados en el desarrollo y las conexiones necesarias para transmitir las señales eléctricas y enviarlas al software desarrollado (disp. Android):*



## Librerías y recursos extras:

*Para la comunicación mediante Bluetooth se utilizó un plug-in público desarrollado en Android Studio llamado BlueUnity, el cual permite la comunicación entre Unity 3D y Android. Este plug-in está documentado en GitHub donde podemos encontrar toda la información necesaria para la incorporación al proyecto deseado. Además, el desarrollador provee una aplicación de test (BluetoothTest) para ejecutar en Android con comunicación bluetooth con Arduino.*

## Futuras mejoras

- *Agregar otros modelos de robot*
- *Movimiento de cada coordenada articular mediante potenciómetro*
- *Menú para configuraciones*
- *Programación de tareas desde el software*
- *Conexión con matlab o similar para ejecutar cinemáticas*
- *Guía de ayuda para el usuario*
- *Mejorar el hardware y reemplazar por joystick con mejor ergonomía*



## Conclusión

*El presente trabajo sirvió para adentrarse en el mundo de la realidad virtual, conocer las herramientas disponibles y las oportunidades que éstas ofrecen. En nuestra breve experiencia con los softwares implementados, podríamos decir que, por un lado, Blender es una excelente y muy potente herramienta de diseño 3D orientada a aplicaciones Gamers o de realidad virtual / mixta.*

*Asimismo, Unity resultó ser muy potente, útil y simple. Posibilitando la incorporación, vinculación y manipulación de trabajos desarrollados en softwares diversos. Dispone de una flexibilidad y amplitud en los desarrollos posibles que nunca antes se experimentaron. Por esto mismo, el software requiere de una gran potencia de cómputo para ejecutar los trabajos que se propongan.*

*Si bien en ambos casos se presentaron algunas dificultades que prolongaron los tiempos de desarrollo, se consiguieron resultados sobresalientes en el marco de proyecto de fin de cátedra, pudiendo dar con el resultado deseado. Se diseñó finalmente una aplicación Android robusta y funcional, con su respectivo control remoto bluetooth. Por gusto y realización personal de los alumnos quedan algunas modificaciones y mejoras que podrían realizarse si se quisiera agregar valor al proyecto y escalarlo a uno de mayores dimensiones.*

## Bibliografía

- ABB IRB1410: <https://new.abb.com/products/robotics/es/robots-industriales/irb-1410>
- Unity Scripting API: <https://docs.unity3d.com/ScriptReference/>
- Vuforia Engine: <https://developer.vuforia.com/>
- BlueUnity: <https://github.com/bentalebahmed/BlueUnity>