

**UNIVERSIDAD POLITECNICA DE VALENCIA**

**ESCUELA POLITECNICA SUPERIOR DE GANDIA**

**Grado en Ing. Sist. de Telecom., Sonido e Imagen**

---

*Departamento de Ingeniería Electrónica*



**UNIVERSIDAD  
POLITECNICA  
DE VALENCIA**



**ESCUELA POLITECNICA  
SUPERIOR DE GANDIA**

# **PRÁCTICA**

**"Generación de un entorno olfativo:**

**Diseño e implementado de un videojuego con feedback  
olfativo con Unity y Arduino"**

**Autores:**

**Peral de León, Jorge**

**Arévalo Jaramillo, Laura  
Marcela**

**Asignatura:**

**Flujo de datos multimedia**

**Tutor/a:**

**Boronat Segui, Fernando**

**GANDIA, 2020-2021**

**ÍNDICE**

<b>1</b>	<b>INTRODUCCIÓN Y OBJETIVOS .....</b>	<b>3</b>
1.1	HARDWARE .....	3
1.2	SOFTWARE.....	4
<b>2</b>	<b>DISEÑO EN UNITY .....</b>	<b>5</b>
2.1	CREACIÓN DEL PROYECTO .....	5
2.2	DESCARGA E IMPORTACIÓN DE ASSETS .....	6
2.3	DISEÑO DEL VIDEOJUEGO .....	7
<b>3</b>	<b>IMPLEMENTACIÓN DE CÓDIGO EN VISUAL STUDIO CODE.....</b>	<b>13</b>
3.1	RESOLUCIÓN DE PROBLEMAS DE COMPATIBILIDAD.....	13
3.2	IMPLEMENTACIÓN DE CÓDIGO EN VS .....	14
3.2.1	Implementación de código para la detección de objetos.....	14
3.2.2	Implementación de código para interactuar con el dispositivo de Arduino .....	17
<b>4</b>	<b>IMPLEMENTACIÓN DE CÓDIGO EN ARDUINO IDE .....</b>	<b>20</b>
<b>5</b>	<b>MONTAJE DEL CIRCUITO.....</b>	<b>24</b>
<b>6</b>	<b>MONTAJE DEL DISPOSITIVO .....</b>	<b>24</b>
<b>7</b>	<b>CREACIÓN DE UN EJECUTABLE DEL JUEGO.....</b>	<b>25</b>
<b>8</b>	<b>ANEXOS .....</b>	<b>26</b>
8.1	ANEXO 1: L293D PIN CONNECTIONS.....	26
8.2	ANEXO 2: DATASHEET DE ARDUINO ELEGOO UNO R3 .....	27
8.3	ANEXO 3: PLANO 3D DEL MÓDULO.....	28





# 1 Introducción y Objetivos




El objetivo de esta práctica es la creación de un videojuego interactivo el cual proporcione al usuario un feedback olfativo al entrar en contacto con ciertos aspecto del juego.

Para lograrlo se realizará el diseño del juego en Unity, haciendo uso de Scripts programadas en Visual Studio (VS) y assets predefinidos y descargados del Asset Store de Unity. Por otro lado se configurará la comunicación entre Unity y la placa de Arduino para habilitar la generación de aromas que se controla con dicha placa.

## 1.1 Hardware

El dispositivo olfativo con Arduino se realizará a partir de los materiales descritos en la Tabla 1. Estos materiales se conectarán entre si mediante el uso de cables Jumper conectados a la tabla de circuito experimental, según la disposición mostrada en la Ilustración 19.

Material	Descripción	AÑADIR LA FOTO
Placa controladora ELEGOO UNO R3		
830 Tie-Points breadboard	Placa para pruebas	
L293D	Controlador de motor de 16 pines	
3.3V/5V output MB102 Breadboard	Adaptador de tensión (de 9V a 6V) para alimentar el controlador	

Esencia (cuyo tamaño sea inferior a 8cm)		
Adaptador de 9V 1A	Adaptador para alimentar el Arduino	
Motor 3 V	Para el ventilador	
Caja para el modulo	Envoltura impresa en 3D	

*Tabla 1 Listado de materiales para el dispositivo de feedback olfativo*

Debido a la situación actual provocada por la COVID-19 en lugar de realizarse un dispositivo que se pueda acoplar a unas gafas de Realidad Virtual (VR), se diseñará un dispositivo unimodular sin la necesidad de contacto físico con los distintos usuarios.

## 1.2 Software

Para la realización de la práctica es necesaria la instalación de los softwares descritos en la Tabla 2, los cuales están incluidos en el fichero del proyecto en el apartado de Software, aunque también son accesibles en los links de las referencias.

Software	Propósito
Unity Hub 2.4.2 <sup>1</sup>	Herramienta de gestión de archivos para la instalación y ejecución de proyectos en Unity.
Unity 2019.4.17f1 <sup>2</sup>	Herramienta de diseño de videojuegos tanto 2D como 3D que permite el uso de código en C#.
VisualStudio <sup>3</sup>	Editor de código
Arduino IDE <sup>4</sup>	Editor de código para la implementación y subida de código en Arduino.

Tabla 2 Lista de Software necesario para la realización de la práctica

## 2 Diseño en Unity

### 2.1 Creación del proyecto

Tras la instalación de los programas *Unity Hub* y *Unity 2019.4.17f1* se puede proceder a la creación de un nuevo proyecto de *Unity*. Para ello primero se ha de ejecutar el programa *Unity Hub* y seleccionar sobre la pestaña *NEW* que se muestra en la ilustración 1.

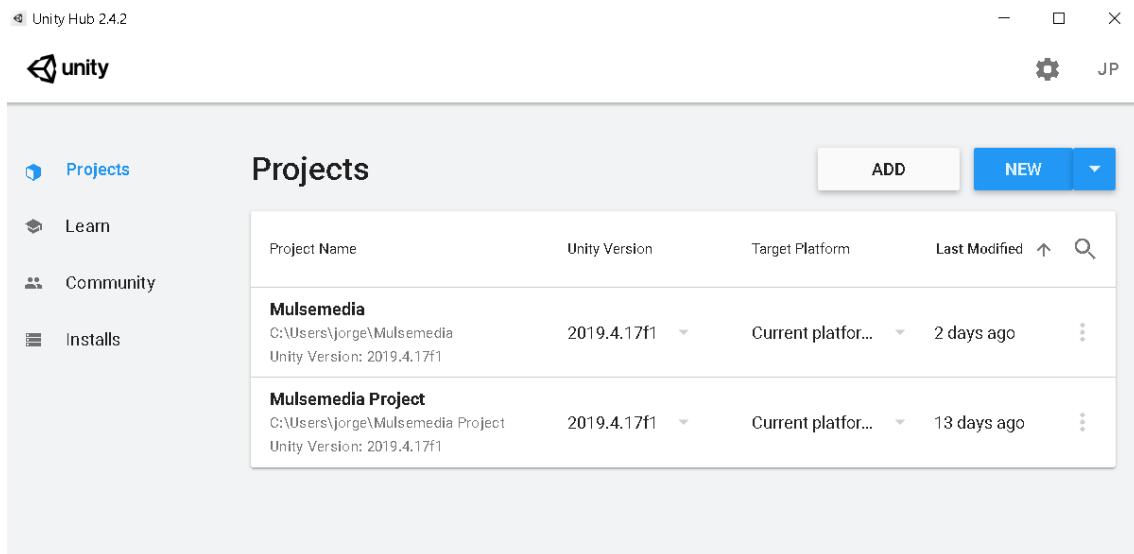


Ilustración 1 Pantalla de Inicio de programa Unity Hub v2.4.2

Debido a la imposibilidad de realizar este proyecto con gafas VR se elegirá el Modelo 3D y se rellenarán los campos de *Project Name* y *Location* tal y como se estipula en la ilustración 2, cambiando <nombre de usuario> por el del alumno.

Una vez completado el paso anterior, seleccionar sobre *CREATE*, lo que creará el proyecto (puede llegar a tardar varios minutos en completarse).

<sup>1</sup> <https://unity3d.com/get-unity/download>

<sup>2</sup> <https://unity3d.com/get-unity/download>

<sup>3</sup> <https://code.visualstudio.com>

<sup>4</sup> <https://www.arduino.cc/en/software>

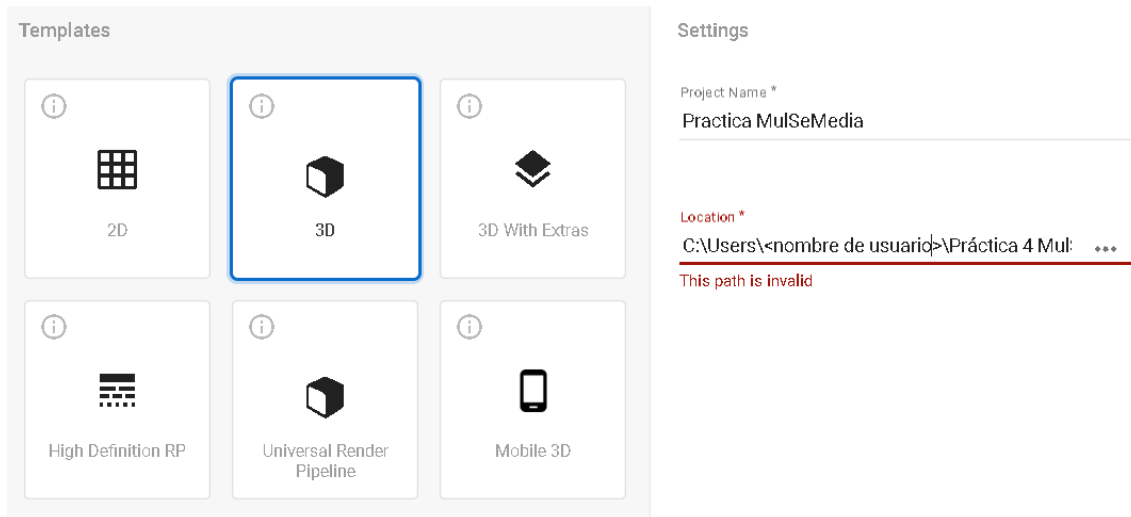


Ilustración 2 Creación de un proyecto en Unity Hub

## 2.2 Descarga e Importación de Assets

Una vez generado el proyecto se deberán descargar los distintos assets necesarios para la creación del proyecto. Para ello seleccionar la pestaña de *Asset Store* que se encuentra en la ventana principal de *Unity* que se muestra en la ilustración 3. Esto abrirá una página integrada dentro de la propia interfaz de *Unity* donde se deberán buscar los siguientes assets, las cuales deberemos comprar e importar:

- *Standard Assets* (para Unity 2018.4)
- *Uduino - Arduino and Unity communication, simple, fast and stable*

Este paso puede llevar varios minutos.

La descarga del asset “*Standard Assets (for Unity 2018.4)*” genera un error *CS0234* debido a que la versión para la que está optimizado es anterior a la versión actual de *Unity*. Este error se resuelve en el apartado *Implementación de Código en Visual Studio Code*.

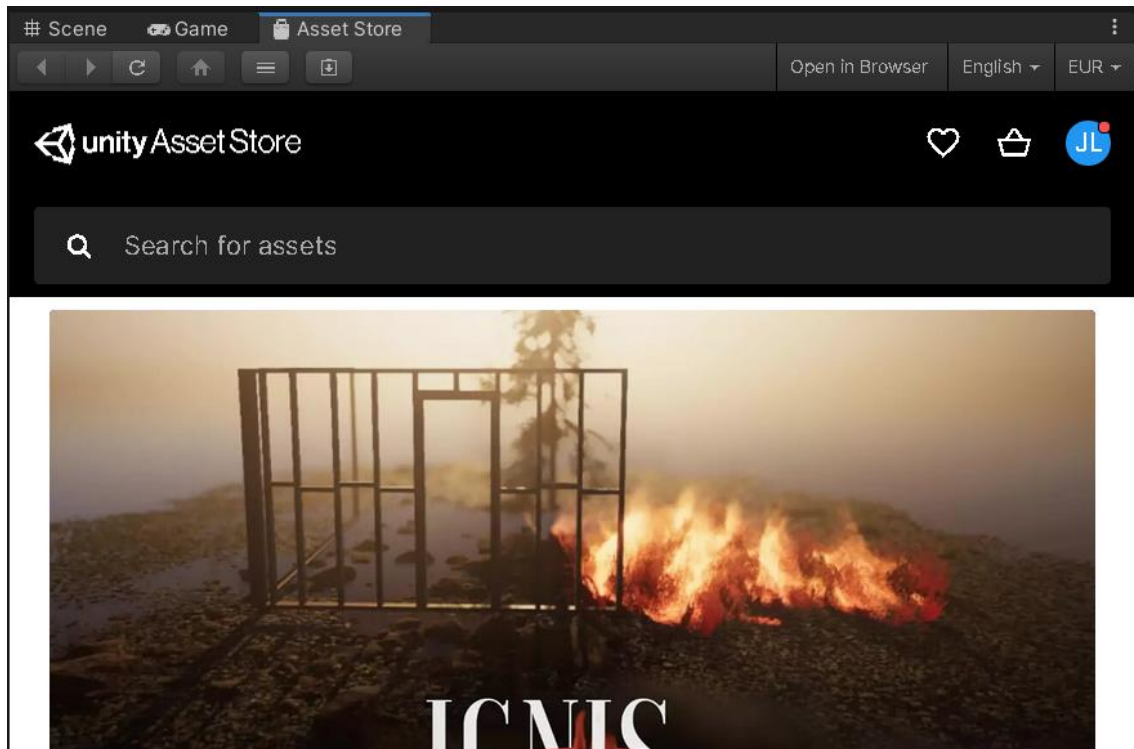


Ilustración 3 Panel de búsqueda de Unity Asset Store

Una vez importados los *Assets* se dará paso a la parte de diseño del videojuego.

## 2.3 Diseño del videojuego

En este apartado se va a explicar cómo se genera el entorno virtual del juego con Unity.

El entorno consiste en crear un habitáculo donde hay un objeto que al seleccionarlo se va a emitir un aroma.

### A) Habitáculo

En primer lugar se creará el habitáculo.

Como se muestra en la Ilustración 4 Creación de un plano en la escena se selecciona la pestaña *Scene* y se hace clic con el botón derecho del ratón sobre el apartado “*Hierarchy*” en la parte izquierda de la pantalla. Aquí seleccionar *3D Object > Plane*, lo que generará el plano donde tomará lugar la escena.

Para aumentar el tamaño del plano, se ha de seleccionar sobre *Plane* en el apartado *Hierarchy*, lo que desplegará las propiedades del plano en la parte derecha de la pantalla (Pestaña *Inspector*).

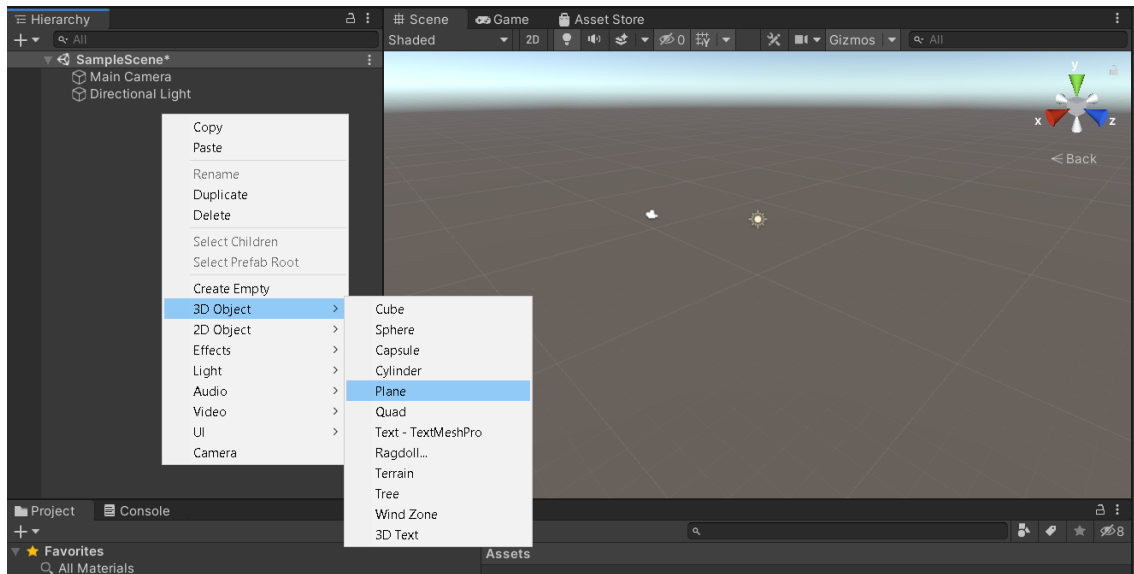


Ilustración 4 Creación de un plano en la escena

Modificar la escala (*Scale*) para los ejes X y Z tal como se muestra en la Ilustración 5.

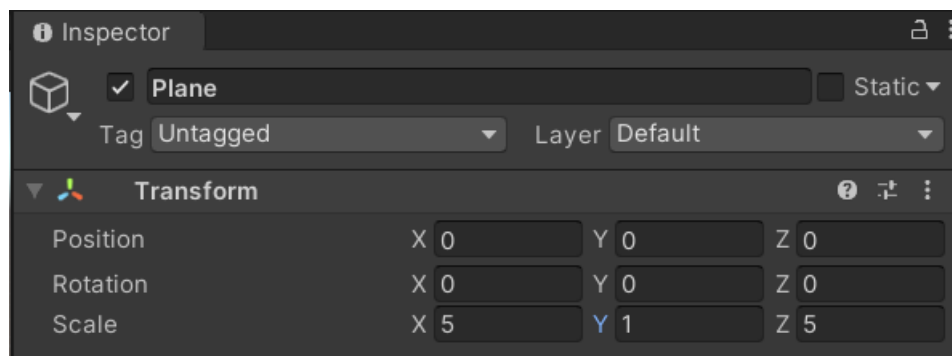


Ilustración 5 Parámetros del plano en Unity

A continuación, seleccionar, en la parte inferior dentro de la pestaña *Project*, la carpeta *Assets*. Se deberá abrir el archivo de los *Prefabs* del *FirstPersonCharacter*. Para ello, se seguirá la ruta *Assets > Standard Assets > Characters > FirstPersonCharacter > Prefabs*, tal como se muestra en la Ilustración 6.

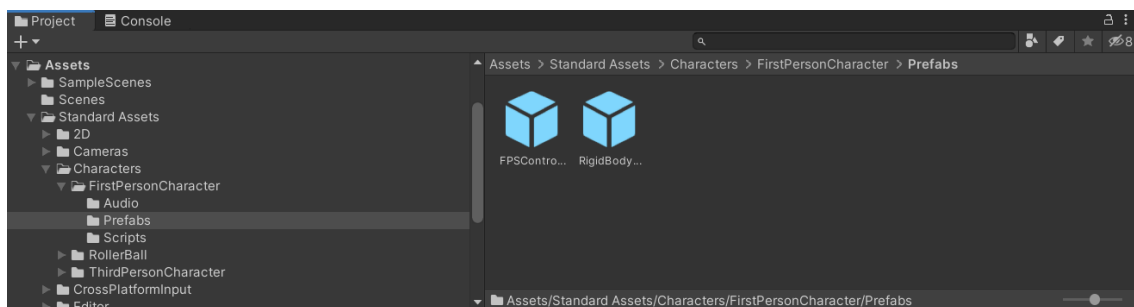


Ilustración 6 Crear un personaje en primera persona desde la pestaña "Project"



Una vez dentro de la carpeta seleccionar y arrastrar el *Prefab FPSController* a la pestaña *Hierarchy*. Esto generará un personaje en primera persona que será el controlado por el usuario.

El siguiente paso que llevar a cabo sería la eliminación del objeto *Main Camera*, para ello seleccionarlo en la pestaña *Hierarchy* y presionar la tecla *Supr* del teclado.

A continuación, se van a generar paredes y a delimitar el plano del escenario virtual. Se realizará un procedimiento similar al de los pasos anteriores.

Primero ir a la carpeta *Assets > Standard Assets > Prototyping > Models* y seleccionar el objeto *WallPrototype08x08x01.fbx* que se muestra en la Ilustración 7, y arrastrarlo a *Hierarchy* un total de 4 veces. Esto generará 4 paredes.

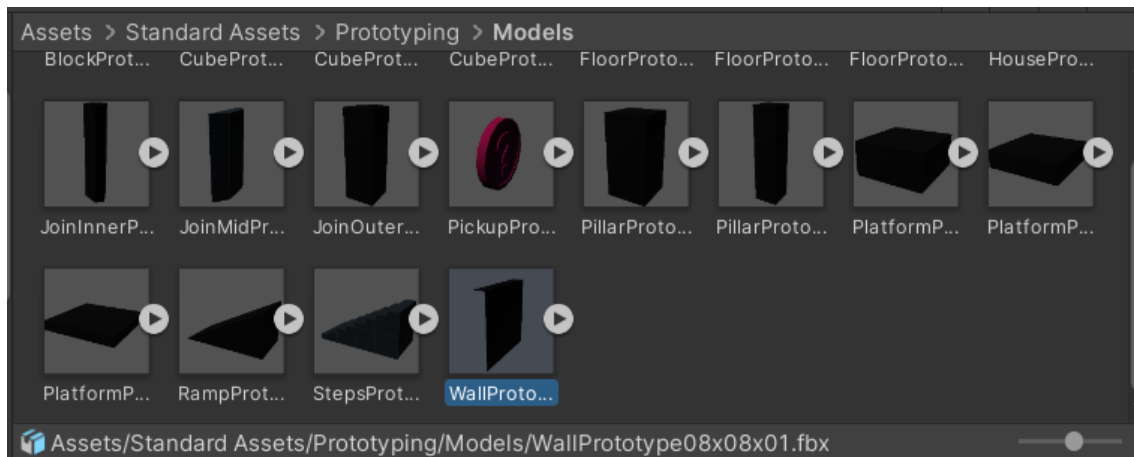


Ilustración 7 Búsqueda de un objeto Pared

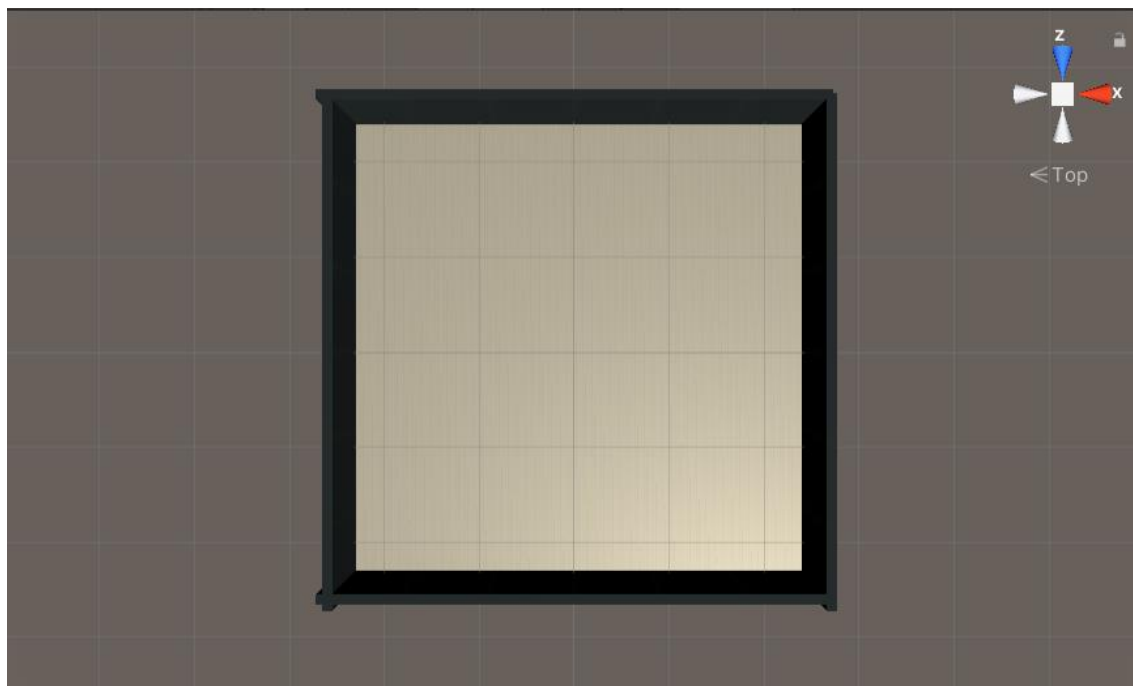


Ilustración 8 Plano en Unity delimitado por paredes

Para colocar las paredes en su posición se deben seleccionar una a una en el apartado “*Hierarchy*” y modificar sus parámetros según se muestra en la tabla 4.

	Position			Rotation			Scale		
Nombre	X	Y	Z	X	Y	Z	X	Y	Z
WallPrototype08x08x01	0	0	24	0	0	0	8.2	1	1
WallPrototype08x08x01(1)	24	0	0	0	90	0	8.2	1	1
WallPrototype08x08x01(2)	-24	0	0	0	90	0	8.2	1	1
WallPrototype08x08x01(3)	0	0	-24	0	0	0	8.2	1	1

Tabla 3 Parámetros de Transformación de las Paredes

#### B) Añadir un objeto al habitáculo

Se incluirá un bloque cuadrado (cubo) en un lateral del habitáculo.

Pulsar botón derecho sobre la ventana *Hierarchy* y seleccionar *3D Object > Cube*. Añadirlo al escenario y modificar sus parámetros de posición a la deseada dentro del plano. Colocarlo en un lateral.

Ir a la carpeta *Uduino* y arrastrar el objeto que se muestra en la Ilustración 9 a la pestaña de *Hierarchy*, lo que generará un objeto que será el que se utilizará para enviar datos a la placa de Arduino.



Ilustración 9 Objeto Uduino

Si no lo está ya, se debería añadir la librería *Uduino* en la carpeta, siguiendo las instrucciones mostradas en el interface de Unity. Por defecto se instala en la carpeta

%USER%\Documents\Arduino\libraries

Aparecerá un mensaje de Error de conexión con Arduino (No Arduino connected).

Para finalizar con el diseño de la escena, se va a incluir un objeto *Canvas* en el que se mostrarán mensaje de texto al usuario (avisos, instrucciones...). Para ello, ir a *Hierarchy*, pulsar ratón derecho y seleccionar *UI > Text-TextMeshPro*. Se deberán modificar las propiedades de este último objeto tal y como se indica en la Ilustración 10.

Una vez incluido, pulsar sobre ‘canvas’ en la ventana de *Hierarchy* y modificar sus parámetros de acuerdo con la parte superior de la Ilustración 10. Para incluir la cámara del personaje pulsar sobre el icono con dos círculos concéntricos que hay a la derecha. Con esto se indica que el canvas donde se van a incluir mensajes de texto estará a 3 metros del personaje, para que los pueda leer bien.

A continuación, modificar los parámetros del objeto *Text-TextMeshPro* de acuerdo con lo indicado en la parte inferior de la Ilustración 10.

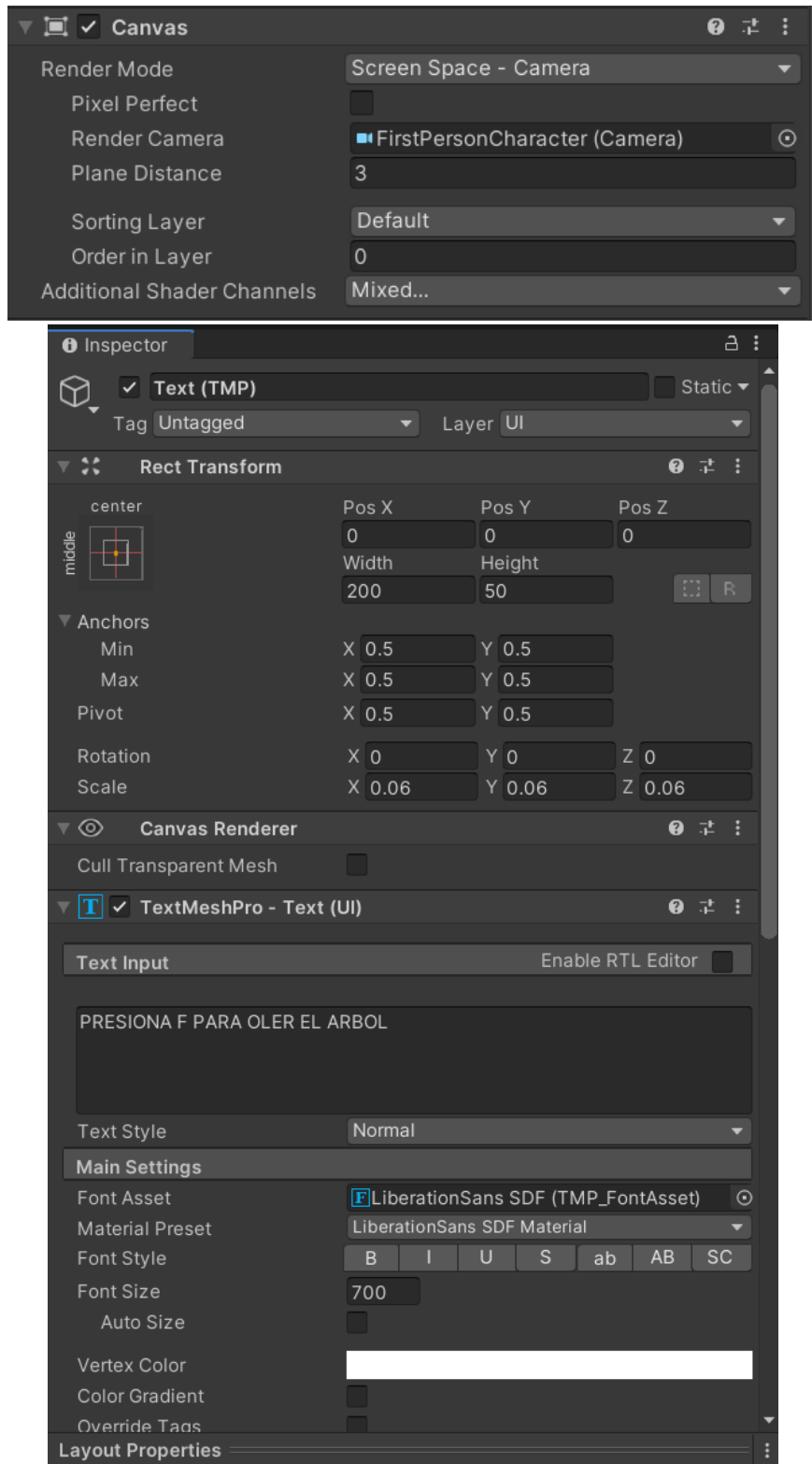


Ilustración 10 Propiedades del objeto Canvas y TextMeshPro

### 3 Implementación de Código en Visual Studio Code

#### 3.1 Resolución de problemas de compatibilidad

Para solucionar los Errores producidos por el desfase del paquete básico primero hay que seleccionar el error con código CS0619. Esta acción abrirá una página de texto en *VisualStudio* en la cual se realizarán los siguientes cambios.

1. En la línea 3 pegar la siguiente línea de código.

```
using UnityEngine.UI;
```

2. En la línea 11, sustituir el código presente por el que se muestra a continuación.

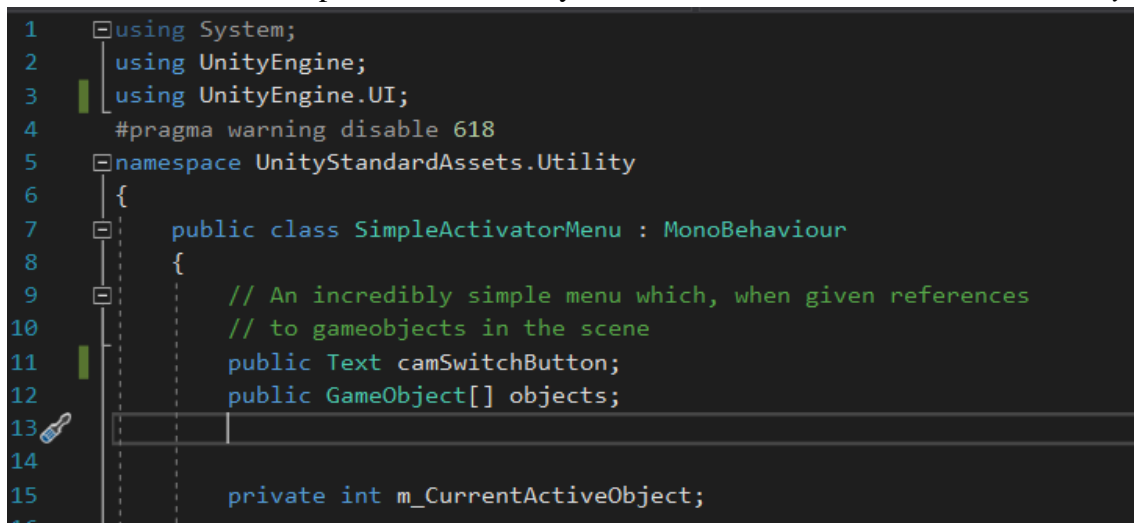
```
public Text camSwitchButton;
```

3. Guardar el fichero utilizando el comando *Ctrl+S*.

Por lo tanto el código quedaría como se muestra en la Ilustración 11.

Una vez de vuelta a *Unity* los errores han desaparecido y se puede probar el juego dándole al botón de *Play*, en la parte superior de la pantalla.

Para salir de este modo pulsar la tecla *Esc* y tras ello volver a seleccionar el botón *Play*.



```
1 using System;
2 using UnityEngine;
3 using UnityEngine.UI;
4 #pragma warning disable 618
5 namespace UnityStandardAssets.Utility
6 {
7     public class SimpleActivatorMenu : MonoBehaviour
8     {
9         // An incredibly simple menu which, when given references
10        // to gameobjects in the scene
11        public Text camSwitchButton;
12        public GameObject[] objects;
13
14
15        private int m_CurrentActiveObject;
16    }
```

Ilustración 11 Solución del Error CS619

## 3.2 Implementación de código en VS

### 3.2.1 Implementación de código para la detección de objetos

A continuación se creará una carpeta en el directorio de assets llamada *Scripts* donde se guardarán los ficheros de código creados a más adelante. Para ello, hacer clic derecho sobre la pestaña *Project*, sobre de la carpeta *Assets* y seleccionar la opción *Create > Folder*. A esta nueva carpeta se le puede dar el nombre que se desee, pero se recomienda un nombre auto descriptivo como *Scripts*.

Una vez realizado esto, seleccionamos la carpeta y dentro de ella, clic derecho *Create > C#Script*. Este último paso lo realizaremos dos veces, llamando a uno de los ficheros *Item* y al otro *HoldFTToInteract*.

A continuación se arrastrarán los ficheros de código de la ventana de *Scripts* (inferior) al objeto al que estarán asignados (en la ventana *Hierarchy*), de la siguiente forma:

Script	Objeto
Item	Cube
HoldFTToInteract	Uduino

Tabla 4 Asignación de los Ficheros de código C# a sus respectivos objetos de la Escena

En caso de dar el error que se muestra en la Ilustración 12, hacer doble clic (izquierdo) sobre el fichero y una vez en VS modificar la línea 5 de la siguiente manera:

```
public class <Nombre del Script que falla> : MonoBehaviour
```

Una vez hecho esto, seleccionar el objeto *Cube* y en la pestaña Inspector, clicar sobre *Layer > Add Layer*, añadiendo un añadiendo un nuevo Layer (por ejemplo *User Layer 8*) con el nombre *Item*, como se muestra en la Ilustración 13. A continuación seleccionar en el objeto *Cube* la nueva capa creada *Layer > Item*.

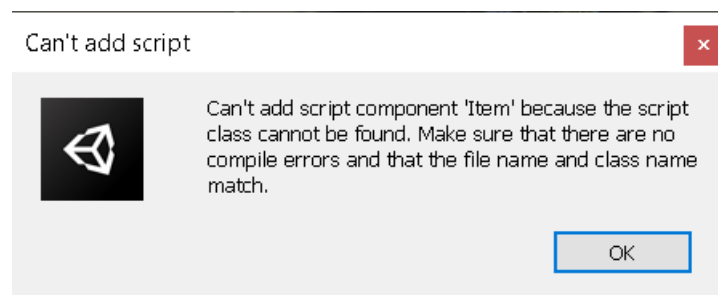


Ilustración 12 Error de creación de fichero donde el nombre de este no coincide con el nombre de la clase

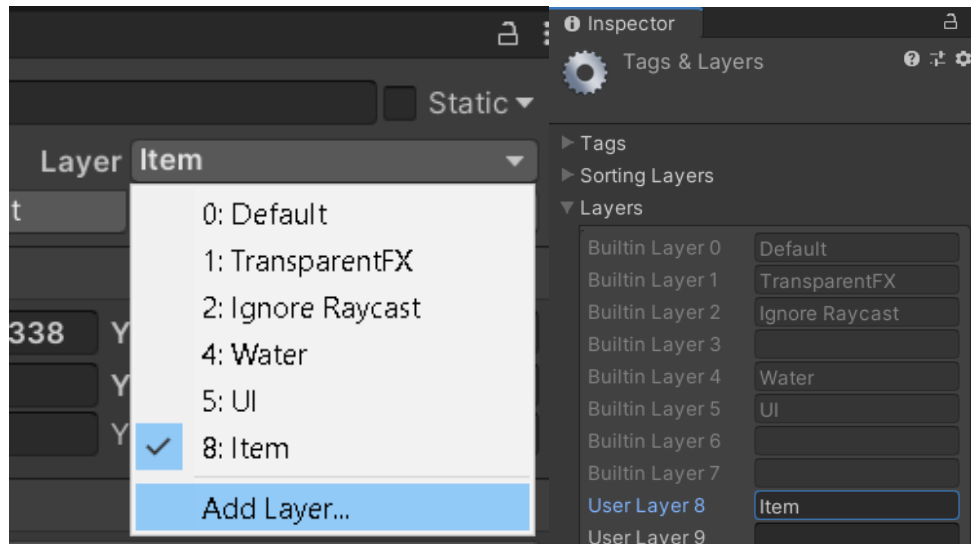


Ilustración 13 Creación de un nuevo User Layer

A continuación, abrir el fichero *HoldFTToInteract* en VS. Para ello hacer doble click sobre el fichero en la ventana de scripts y se abrirá el VS o el editor seleccionado por defecto. En esta práctica únicamente se modificará el fichero *HoldFTToInteract* ya que el fichero *Item* tiene el sólo propósito de identificar al objeto *Cube*.

Sustituir los paquetes que se van a utilizar en las 3 primeras líneas de código por lo siguiente:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro; //Paquete que utiliza TextMeshPro
```

Copiar el siguiente código dentro de la clase *HoldFTToInteract*:

```
public class HoldFTToInteract : MonoBehaviour
{
    /*El código a continuación permite elegir de forma dinámica los distintos
    componentes que toman parte en la clase, sin necesidad de saber su nombre*/
    [SerializeField]
    [Tooltip("Se trata de la cámara principal pero se referencia, para evitar
    llamadas a Camera.main")]
    private Camera camera;
    [SerializeField]
    [Tooltip("La capa donde se encuentra el objeto con el que se desea
    interactuar... Item")]
    private LayerMask layerMask;
    [SerializeField]
    [Tooltip("Tiempo que se desea que se mantenga pulsada la tecla para llevar
    a cabo la función.")]
    private float pickupTime = 2f; //2 segundos
    [SerializeField]
    [Tooltip("Objeto de texto que se desea que le aparezca al jugador")]
    private TextMeshProUGUI itemNameText;

    private Item itemBeingPickedUp; //identificador del item a coger
```

```

private float currentPickupTimerElapsed;

int turnable = 0; /*indicador de si se ha pulsado y si se puede volver a
pulsar 0=no; 1=SI*/

// Update is called once per frame
void Update()
{
    SelectItemBeingPickedUpFromRay(); /*llamada a función de detección de
objetos*/

    if (HasItemTargetted()) /*Si se esta apuntando al objeto
    {
        itemNameText.gameObject.SetActive(true); /*Activar el texto
        if (Input.GetKeyDown(KeyCode.F)) /*si se pulsa la tecla F
        {
            Debug.Log("pilla la F");
            turnable = 1;
        } /*if F is pressed
        if (turnable == 1)
        {
            //Aquí se implementará la parte de comunicación con arduino
        } /*turnable
        else
        {
            currentPickupTimerElapsed = 0f;
        } /*else

    } /*if HasItemTargetted
    else
    {
        itemNameText.gameObject.SetActive(false); /* el texto esta
desactivado (no se ve)*/
        currentPickupTimerElapsed = 0f;
    } /*else

} /*Update

private bool HasItemTargetted()
{
    return itemBeingPickedUp != null;
} /*HasItemTargetted

private void SelectItemBeingPickedUpFromRay()
{
    Ray ray = camera.ViewportPointToRay(Vector3.one / 2f);
    Debug.DrawRay(ray.origin, ray.direction * 2f, Color.red);

    RaycastHit hitInfo;
    if (Physics.Raycast(ray, out hitInfo, 2f, layerMask))
    {
        var hitItem = hitInfo.collider.GetComponent<Item>();

        if (hitItem == null)
        {
            itemBeingPickedUp = null;
        } /*if
        else if (hitItem != null && hitItem != itemBeingPickedUp)
        {
            itemBeingPickedUp = hitItem;
            itemNameText.text = "Hold F to smell the tree";
        }
    }
}

```



```

        }//else if
    }//if Raycast hits Item
    else
    {
        itemBeingPickedUp = null;
    }//else
}// SelectItemBeingPickedUpFromRay()

}//( )

```

Este código permite la detección y selección de un *Item* mediante la creación de un vector (Ray) de 2 metros en el centro del campo de visión del usuario, lo que permite que en el momento que un objeto, dentro de la capa *Item*, se encuentre en la trayectoria de dicho vector se pueda guardar el mismo dentro de las propiedades de la clase. Esto hace que el texto que avisa al jugador sobre las instrucciones a seguir aparezca indicándole que se ha habilitado la función olfativa.

A continuación se probará el funcionamiento del escenario diseñado y que al acercarse al objeto cubo menos de 2 metros, se visualiza un mensaje de texto.

Antes se asegurará que en el inspector del *FirstPersonCharacter*, en el *Script* se han habilitado las siguientes opciones.

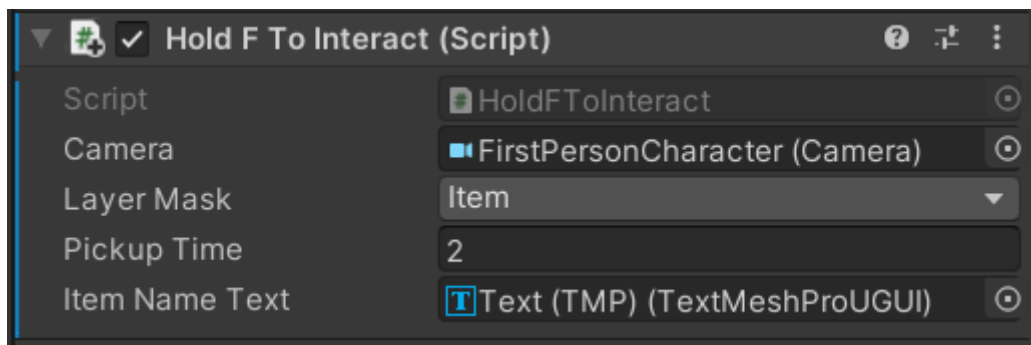


Ilustración 14 Configuración de función Hold F To Interact en Unity

### 3.2.2 Implementación de código para interactuar con el dispositivo de Arduino

Abrir el script *HoldFTToInteract* y añadir al principio del fichero

```
using Uduino;
```

, y, a continuación, pegar el siguiente código dentro de la función indicada en un comentario en el propio código del apartado anterior.

```

        UduinoManager.Instance.sendCommand("turnOn");//envia el comando
        turnOn a la placa de Arduino conectada
        Debug.Log("hasta turnOn tambien tira");
        turnable = 0;
        Debug.Log("turnable está a" + turnable);

```

Este código espera a que el jugador pulse la tecla F para mandar un comando a la placa de *Arduino* para que ejecute una función implementada en *Arduino IDE*.

El código final quedaría de la siguiente manera:

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

public class HoldFToInteract : MonoBehaviour
{
    /*El código a continuación permite elegir de forma dinámica los distintos
    componentes que toman parte en la clase, sin necesidad de saber su nombre*/
    [SerializeField]
    [Tooltip("Se trata de la cámara principal pero se referencia, para evitar
    llamadas a Camera.main")]
    private Camera camera;
    [SerializeField]
    [Tooltip("La capa donde se encuentra el objeto con el que se desea
    interactuar... Item")]
    private LayerMask layerMask;
    [SerializeField]
    [Tooltip("Tiempo que se desea que se mantenga pulsada la tecla para llevar
    a cabo la función.")]
    private float pickupTime = 2f;//2 segundos
    [SerializeField]
    [Tooltip("Objeto de texto que se desea que le aparezca al jugador")]
    private TextMeshProUGUI itemNameText;

    private Item itemBeingPickedUp;//identificador del item a coger
    private float currentPickupTimerElapsed;

    int turnable = 0; /*indicador de si se ha pulsado y si se puede volver a
    pulsar 0=no; 1=SI*/

    // Update is called once per frame
    void Update()
    {
        SelectItemBeingPickedUpFromRay(); /*llamada a función de detección de
        objetos*/

        if (HasItemTargetted()) /*Si se está apuntando al objeto
        {
            itemNameText.gameObject.SetActive(true); /*Activar el texto
            if (Input.GetKeyDown(KeyCode.F)) /*si se pulsa la tecla F
            {
                Debug.Log("pilla la F");
                turnable = 1;
            } /*if F is pressed
            if (turnable == 1)
            {
                UduinoManager.Instance.sendCommand("turnOn"); /*envía el comando
                turnOn a la placa de Arduino conectada*/
                Debug.Log("hasta turnOn tambien tira");
                turnable = 0;
                Debug.Log("turnable está a" + turnable);
            } /*turnable
        }
    }
}
```

```

        else
        {
            currentPickupTimerElapsed = 0f;
        } //else

    } //if HasItemTargetted
    else
    {
        itemNameText.gameObject.SetActive(false); /* el texto esta
desactivado (no se ve) */
        currentPickupTimerElapsed = 0f;
    } //else

} //Update

private bool HasItemTargetted()
{
    return itemBeingPickedUp != null;
} //HasItemTargetted

private void SelectItemBeingPickedUpFromRay()
{
    Ray ray = camera.ViewportPointToRay(Vector3.one / 2f);
    Debug.DrawRay(ray.origin, ray.direction * 2f, Color.red);

    RaycastHit hitInfo;
    if (Physics.Raycast(ray, out hitInfo, 2f, layerMask))
    {
        var hitItem = hitInfo.collider.GetComponent<Item>();

        if (hitItem == null)
        {
            itemBeingPickedUp = null;
        } //if
        else if (hitItem != null && hitItem != itemBeingPickedUp)
        {
            itemBeingPickedUp = hitItem;
            itemNameText.text = "Hold F to smell the tree";
        } //else if
    } //if Raycast hits Item
    else
    {
        itemBeingPickedUp = null;
    } //else
} // SelectItemBeingPickedUpFromRay()

} //()

```

## 4 Implementación de Código en Arduino IDE

Conectar la placa de *Arduino* mediante un cable USB-B.

Abrir el software *Arduino IDE*.

Crear un nuevo fichero y pegar el siguiente código.

```
#define ENABLE 5 //Define el PIN 5 con el nombre ENABLE
#define DIRA 3//Define el PIN 3 con el nombre DIRA (gira a la derecha)
#define DIRB 4//Define el PIN 4 con el nombre DIRB (gira a la izquierda)
#include<Uduino.h>
Uduino uduino("FanBoard"); //indica el nombre con el que se reconocerá la placa desde Unity. Muy útil si se desean conectar varias de ellas.

void setup()
{
    //Se asigna el modo de cada Pin. Todos como salida
    pinMode(ENABLE, OUTPUT);
    pinMode(DIRA, OUTPUT);
    pinMode(DIRB, OUTPUT);
    Serial.begin(9600); //Avisa a Arduino que el intercambio de datos se realizara con un bit rate de 9600 bits/s

    uduino.addCommand("turnOn", turnOnFan);//Asocia al comando "turnOn" recibido desde Unity la función turnOnFan, definida más abajo
}

void turnOnFan()
{
    digitalWrite(DIRA, LOW);
    digitalWrite(ENABLE, HIGH); //enable on
    digitalWrite(DIRB, HIGH); //one way

    delay(5000);//delay de 5s
    digitalWrite(ENABLE, LOW);
}

void loop()
{
    uduino.update();
    delay(10);
    //mira la información que se recibe desde Unity cada 10 ms
}
```

Ir al apartado *Herramientas >Puerto* y seleccionar el puerto en el que se encuentre conectado el dispositivo de *Arduino* como se muestra en la ilustración 15. Tras ello clicar en la flecha de la parte superior que se muestra en la ilustración 14.



*Ilustración 15 Botón de Subir en Arduino IDE*

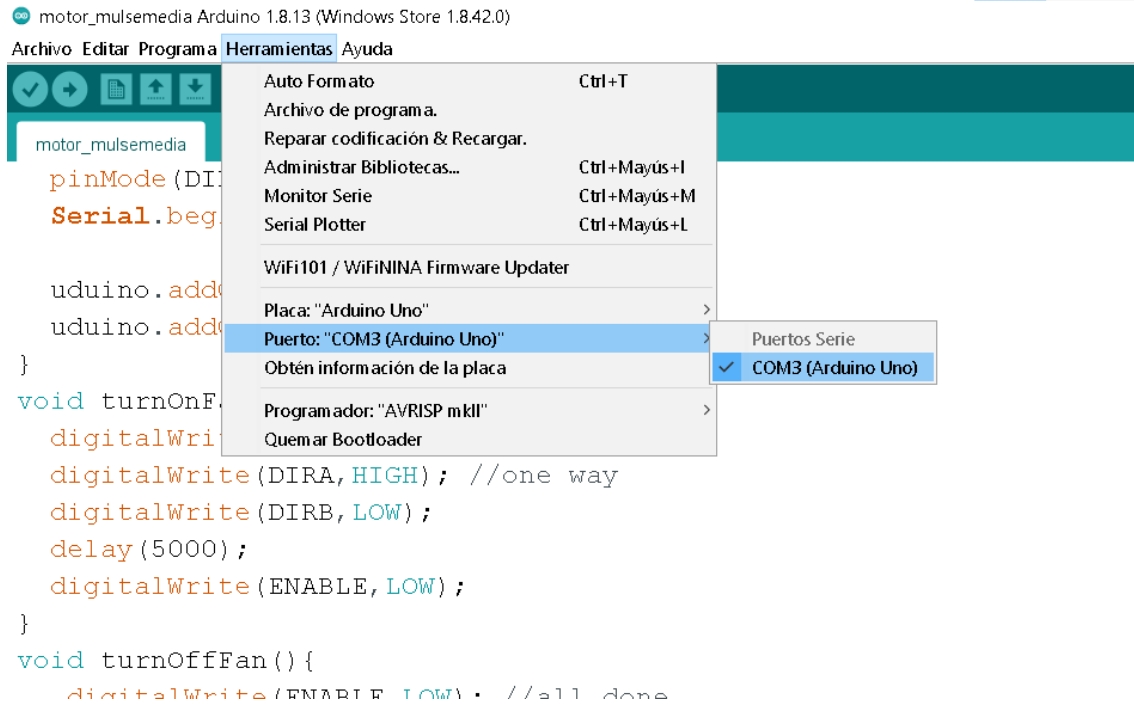


Ilustración 16 Selección de Puerto Serie en Arduino IDE

Para conectar Arduino con Unity, basta con seleccionar el objeto *Uduino* y en la pestaña *Inspector*, clicar botón izquierdo sobre el botón *Fix Now* (Ilustración 17)



Ilustración 17 Cambio de configuración del fichero Unity para poder trabajar con Uduino

Ir al inspector de Uduino y pulsar sobre *Discover Ports*. Deberá aparecer algo parecido a lo mostrado en la Ilustración 18.

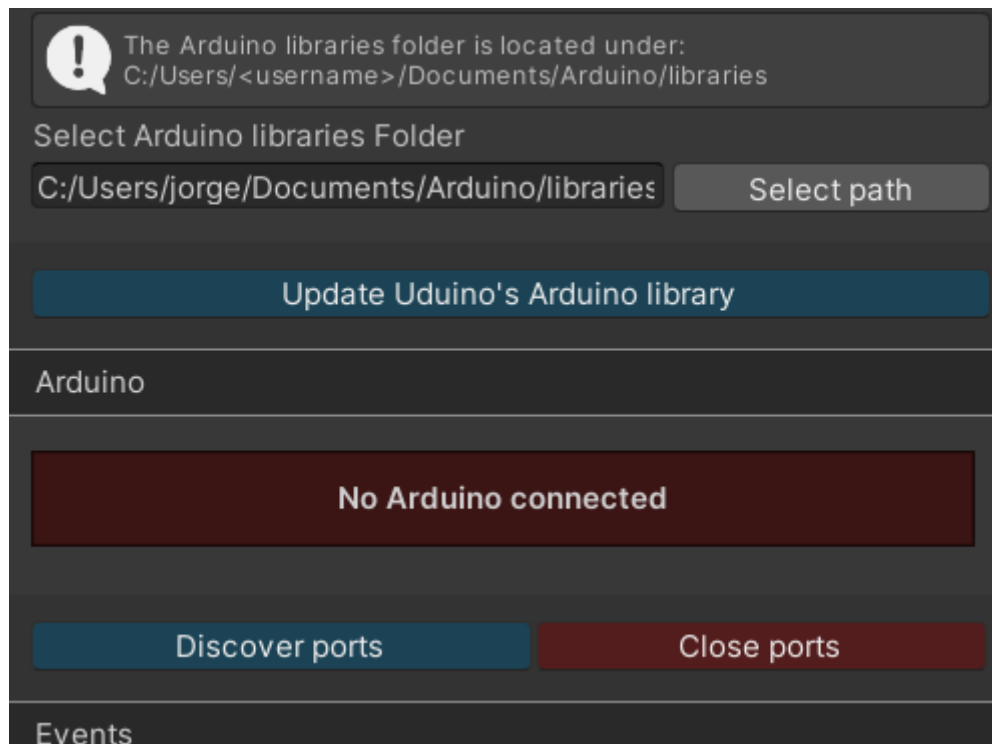


Ilustración 18 Ventana Uduino Discover Ports

## 5 Montaje del circuito

Para el montaje del circuito realizarlo como se muestra en la ilustración 16.

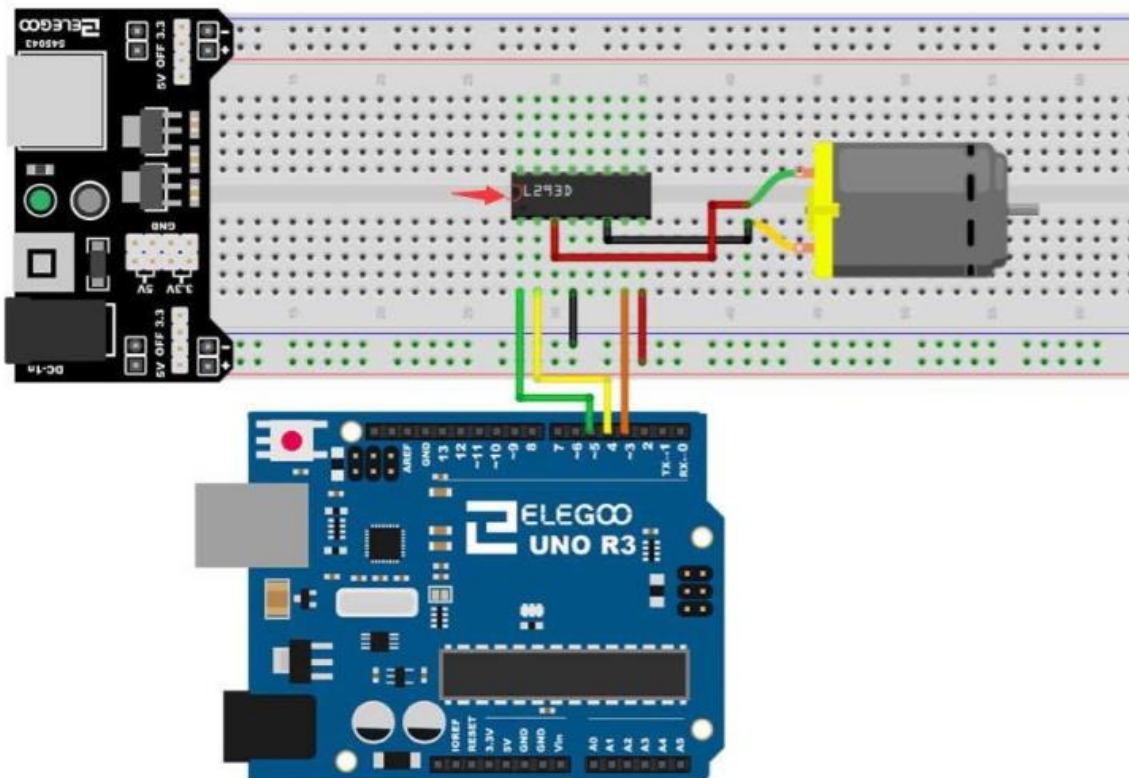


Ilustración 19 Montaje de un circuito con Arduino para el funcionamiento de un motor de 5V. Fuente: <https://docplayer.es/docs-images/93/111676505/images/146-0.jpg>

Conectar el módulo de adaptación de corriente *MB102*.

Las conexiones se realizarán según la tabla 5 teniendo como referencia los pines del *L293D* mostrados en el *Anexo 1*.

Extremo A	Extremo B
Pin 5 de Arduino	Pin ENABLE 1 del L293D
Pin 4 de Arduino	Pin INPUT 1 del L293D
V- del MB102	Pin GND del L293D
Pin 3 de Arduino	Pin INPUT 2 del L293D
V+ del MB102	Pin Vs del L293D
Pin OUTPUT 1 del L293D	V+ del motor
Pin OUTPUT 2 del L293D	V – del motor

Tabla 5 Esquema de conexiones del cableado

## 6 Montaje del dispositivo

Para finalizar, bastaría con el montaje colocar la esencia en el interior del modulo que se muestra en el *Anexo 3*, y posicionar el ventilador en la dirección en la que se encuentra la esencia.



## 7 Creación de un ejecutable del juego

Para realizar una versión operativa del juego, bastaría con seleccionar *File>Build Settings...* como se muestra en la Ilustración 17 o pulsar las teclas *Ctrl+Shift+B*. Tras ello pulsar *Build* en la pestaña *Build Settings*. Tras unos minutos se habrá generado una versión completamente funcional y actualizable en el caso de que los ejecutables posteriores se realicen en la misma carpeta. Al actualizarlos el tiempo de espera es mucho menor.

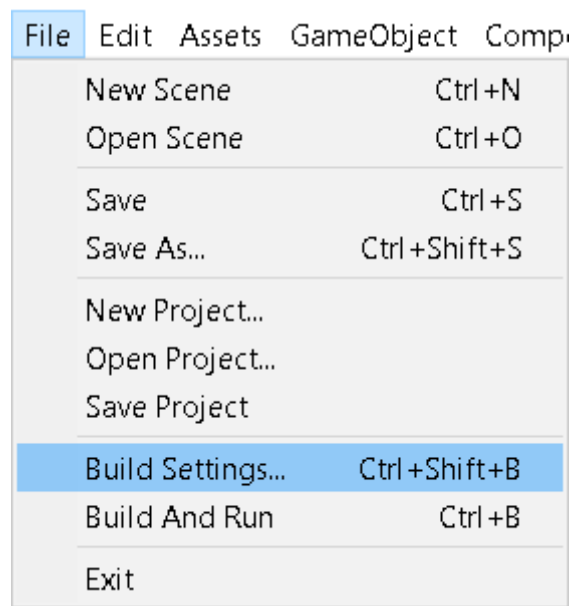


Ilustración 20 Menú en Unity para la generación de un ejecutable

Esto permite que el usuario que desee probarlo no tendrá acceso al código fuente, y, por lo tanto, no será capaz de cambiar parámetros. Además de hacer innecesaria la instalación de *Unity* para el disfrute de este.

## 8 Anexos

### 8.1 Anexo 1: L293D Pin Connections

#### PIN CONNECTIONS (Top view)

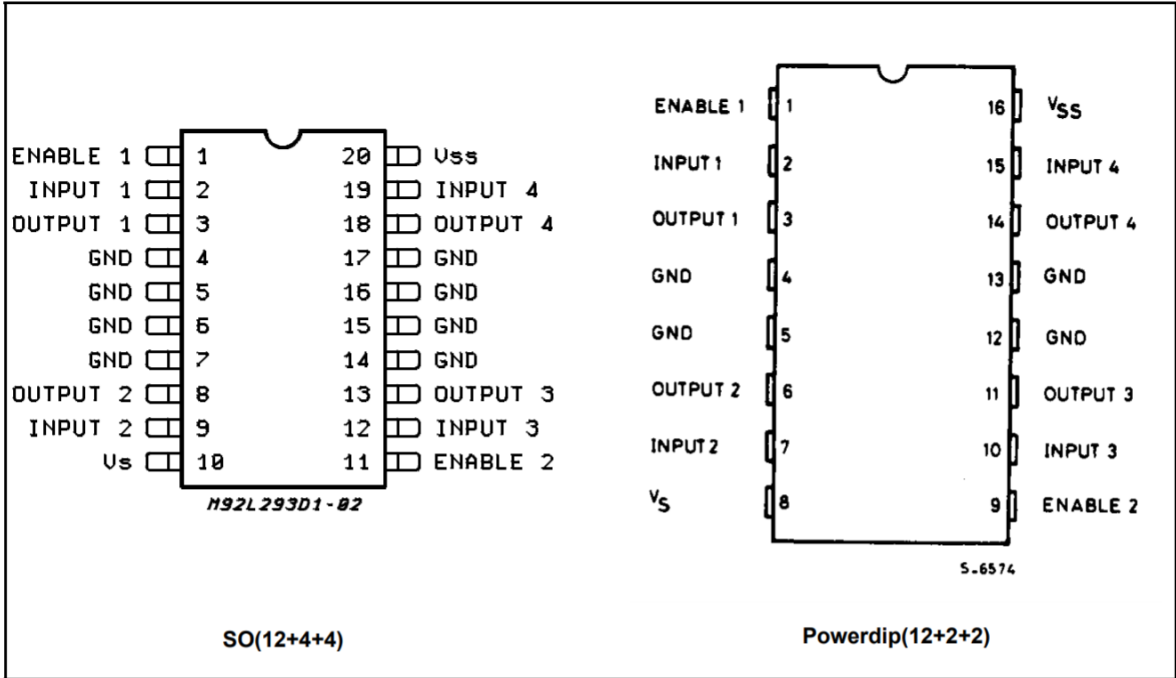
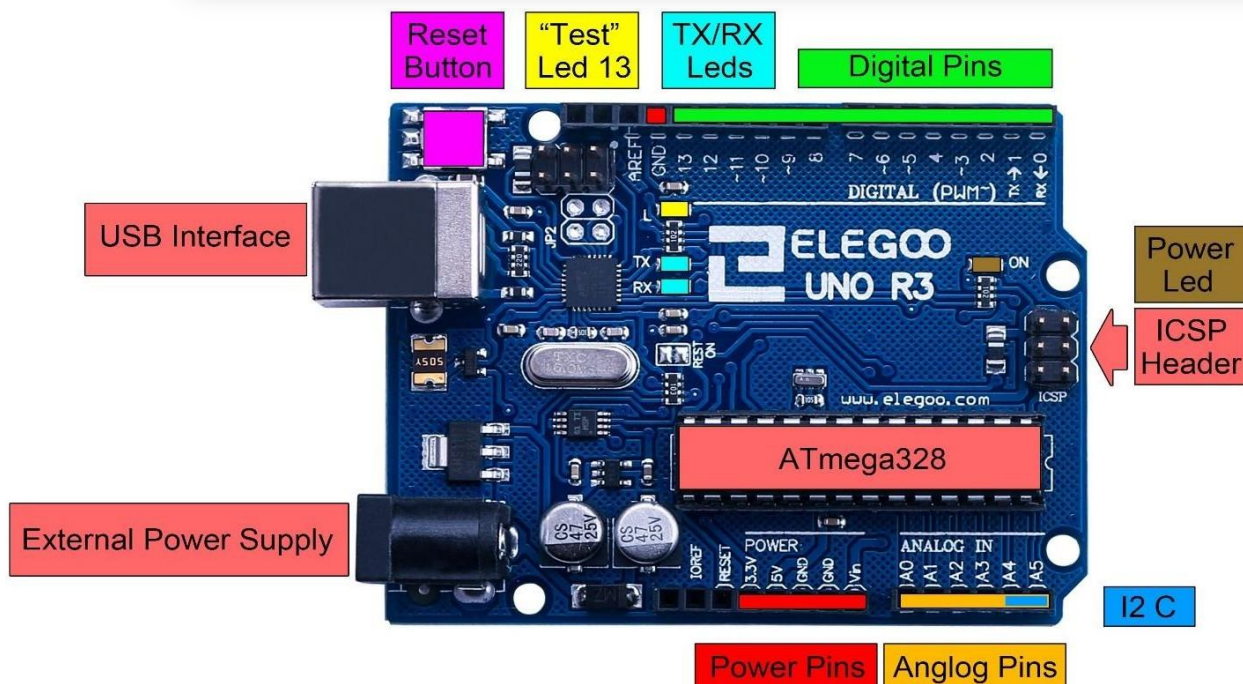


Ilustración 21 Esquema de los 16 pines del controlador L293D. Fuente: <https://pdf1.alldatasheet.com/datasheet-pdf/view/89353/TI/L293D.html>

## 8.2 Anexo 2: Datasheet de Arduino ELEGOO UNO R3



S	
Microcontroller	ATmega328
Operating Voltage	5 V Input Voltage (recommended) 7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14(of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHZ
T	



### 8.3 Anexo 3: Plano 3D del Módulo

