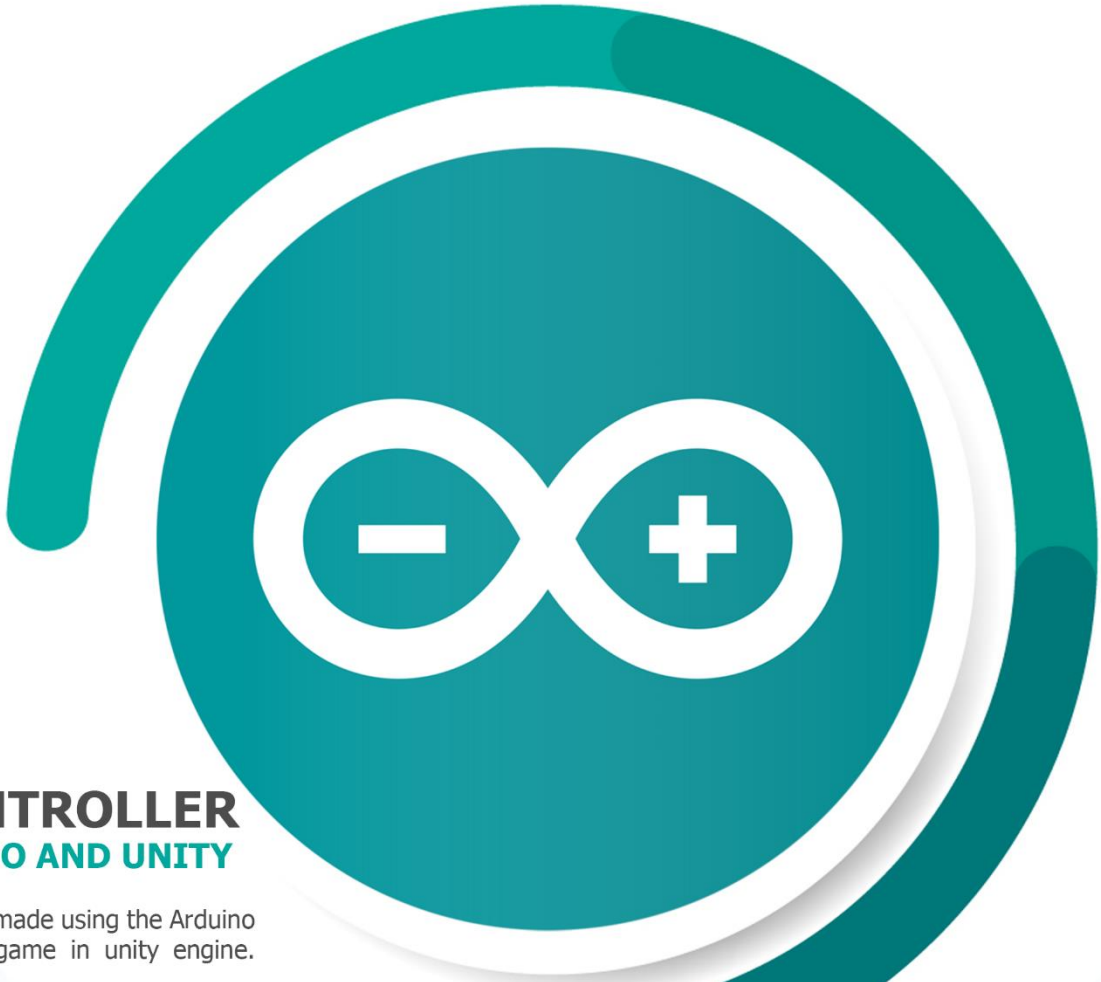
COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY OF ADEN FACULTY OF ENGINEERING



GAME CONTROLLER

USING ARDUINO AND UNITY

This project is a device made using the Arduino that controls a video game in unity engine.

Supervised by Dr. **KHALED ABOOD**

TEAM - B4CS/E

Course Project 1

EMAD ARIF 175521

SAEED AHMED 175508

ALI AHMED 175517

ALI ABDULLA 175544

AMGAD SAEED 135571

2020 - 2021

CONTENTS

FIGURES

TABLES

ACNOWLEDGMENT

First and from the bottom of our hearts we thank God Almighty for giving us the strength to complete this project, then we thank Dr. Khaled Abood - the project supervisor - for providing all the help for us to complete this project to the fullest.

On the other hand, we thank all the teachers who taught us and gave us valuable information as we benefited from it well, and we hope that this project is an appreciation of their efforts and patience with us.

We, the students of the Department of Computer Science and Engineering, designed this project in an attempt to present something new, and on the other hand, try to understand how devices work with software, and we hope that this project is only an introduction to larger projects in the future, God willing.

TEAM MEMBERS

- Emad Aref Saleh
  Register No: 175521

- Saeed Ahmed Mohammed
  Register No: 175508

- Ali Ahmed Saleh Nesher
  Register No: 175517

- Ali Abdulla Alawi Saleh
  Register No: 175544

- Amgad Saeed Abdo
  Register No: 135571

CHAPTER 1: INTRODUCTION

In light of the great advancement in technology, almost everything has become highly interactive with the electronic environment in a different way, such as advanced home lighting control, such as changing its color to controlling the intensity of its brightness via a controller or even via a mobile phone, instead of using the traditional operating switches suspended in the walls of our house.

Because we love games, we wanted to design an interactive device so that this device (the controller) controls a game on the computer through another device instead of using the mouse and keyboard, so we did research and found that the Arduino is the best solution to implement this project.

PROJECT SCOPE

After extensive research on the topic of controllers, we found that there are 4 basis points that we must look at in order to be able to implement the project, which are the following: -

Controllers Design Principles: Controllers are often physical hardware devices that have software programs, as these programs translate and transfer data between two different environments by creating a specific format that enables the two extremes to exchange data.

Mostly, controllers have a set of buttons and measuring tools that are interactive and accurate, enabling the user to interact with the desired thing with great accuracy.

Since we want to design a device that controls a game, we found that we try to design one of the famous controllers as a first trial of the project.

Of course, the controller will contain a set of buttons, in addition to a joystick, and finally the motion and rotation sensor, and as a start, we will try to implement this project in this way.

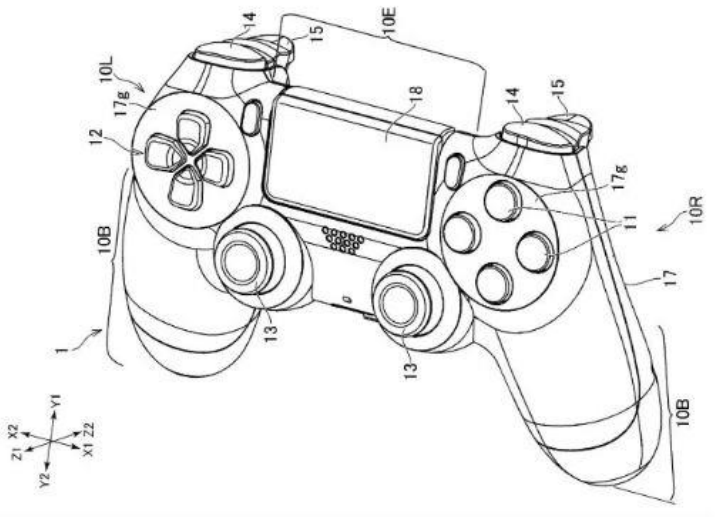


Figure 1: PS4 controller

Making Games Using Unity: In order to be able to fully design our project, we had to learn how to design games using the Unity engine, which is an easy engine and has many online lessons. Our goal while learning to design games was to delve into us in matters such as multi-processing, as well as transfer and reception of data between different platforms.

The programming in the Unity engine is based on the C # language, so it was easy to learn.

Figure 2: C# code in unity

Making Devices Using Arduino: At first, we searched on the Internet for the possibility of designing the controller and found that we will need a set of electronic parts that can be programmed into the Arduino. We purchased the parts from the market after we confirmed that they were required.

We also bought the rest of the things such as wires, Breadboard and some other things such as adhesive tape so that we can install these pieces on the breadboard well. On the other hand, the Arduino is a new environment for us, so we took a course on dealing with the Arduino environment and found that programming in it is based on C and C ++ languages, but we only studied the environment of the Arduino system and libraries because this fulfills our need

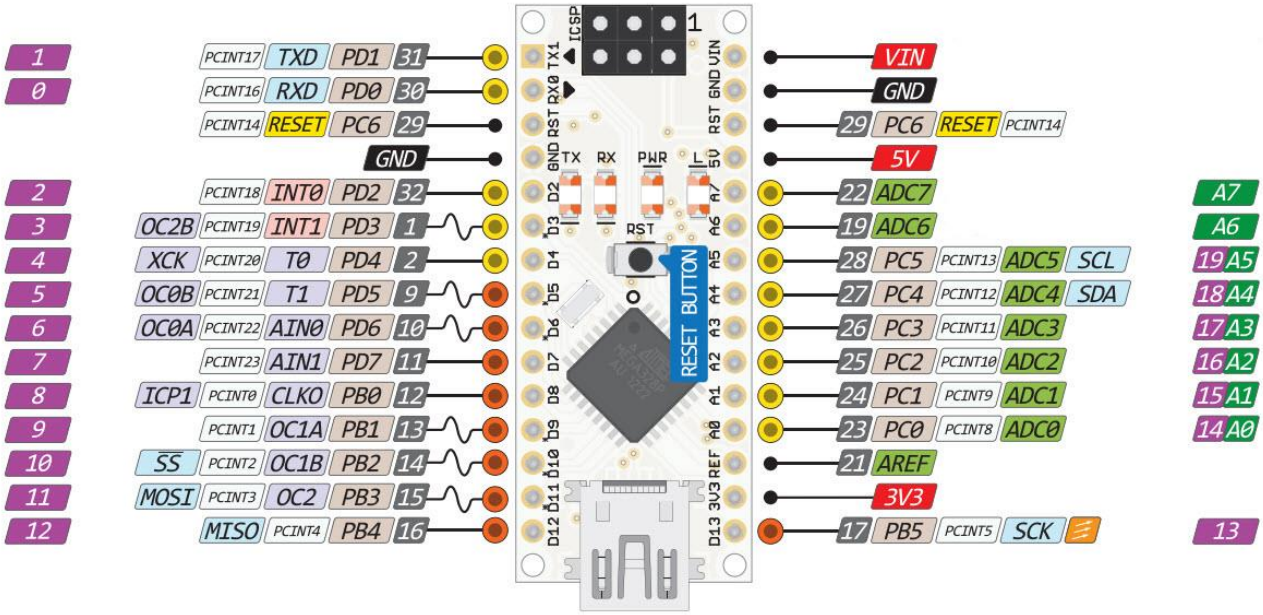


Figure 3: Arduino Nano sheet

Data Transfer Between Unity & Arduino: While we searched for a way to transfer data between two different devices or two different environments, we found that there are several ways in which we can make the two environments exchange information. The first method is to transfer data by wire via the Serial Port in case the two environments support this type, and the second method is to transfer data wirelessly via Bluetooth or Wi-Fi.

Figure 4: Serial Port, Bluetooth and WiFi module

All of these methods have advantages and disadvantages in transferring data in addition to their cost. We finally settled on using the Serial Port to test the project and settled on it.

PROBLEMS

With the advancement and development of technologies, the need to create more interactive environments appeared than before, the VR glasses appeared, which made a quantum leap in the world of games and technology, giving the user the opportunity to see virtual environments in a realistic way while giving him devices that help to control this environment more effectively and realistically.

That is why we wanted to do this project as an attempt from us to understand how we can make a control device as a first idea, and maybe in the future we will develop it more or make it an open system so that everyone can easily design their own controllers.

This study solves the problems of traditional controllers, giving developers - who may like to design their own - the opportunity to design a custom controller in an integrated environment.

OBJECTIVES

- Design a controller using the Arduino and it's components.
- Programming the data transmission system between the Unity engine and Arduino.
- Creating a specific format to encapsulate the data in it so that it can be interpreted between the two environments.
- Make the programming environment easy to use.
- Avoid losing data.
- Design a game that takes advantage of this controller.

CHAPTER 2: REQUIREMENTS

HARDWARE REQUIREMENT

Components:

5 Push Buttons

A push-button (also spelled pushbutton) or simply button is a simple switch mechanism to control some aspect of a machine or a process.

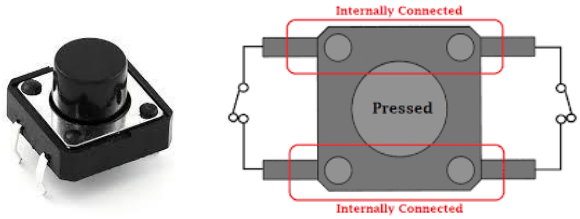


Figure 5: 4 Legs push button

Joystick

An analog stick (or analogue stick in British English), sometimes called a control stick or thumb stick, is an input device for a controller (often a game controller) that is used for two-dimensional input and it has a button.

Figure 6: Joystick X,Y and SW

MPU5060

The MPU6050 is a Micro Electro-Mechanical Systems (MEMS) which consists of a 3-axis Accelerometer and 3-axis Gyroscope inside it. This helps us to measure acceleration, velocity, orientation, displacement and many other motion related parameter of a system or object.

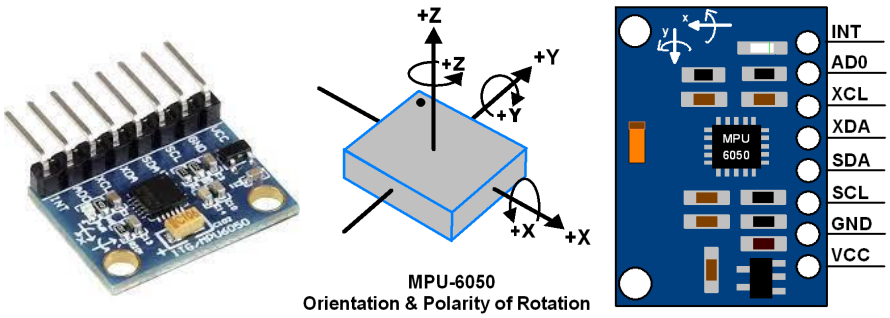


Figure 7: MPU6050 sheet for arduino

5 10KΩ Resistors

A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses

Figure 8: 10KΩ Resistors

Arduino Nano

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one.

Figure 9: Arduino Nano

A Bunch of wires

We need the wires in this case to weld them with all the elements in order to make the current flow correctly without problems or signal loss.

Figure 10: Wires

Computer:

- CPU: Core I5 7th.
- Ram 4GB DDR3 (As Minimum).
- Hard Disk: 300gb (As Minimum).
- External Graphic Card: 2gb (As minimum for games).
- Monitor.
- Mouse & Keyboard

SOFTWARE REQUIREMENT

Unity Game Engine

Unity is a cross-platform game engine developed by Unity Technologies, first announced and released in June 2005 at Apple Inc.'s Worldwide Developers Conference as a Mac OS X-

exclusive game engine. The engine has since been gradually extended to support a variety of desktop, mobile, console and virtual reality platforms. It is particularly popular for iOS and Android mobile game development and used for games. The engine can be used to create three-dimensional (3D) and two-dimensional (2D) games, as well as interactive simulations and other experiences. The engine has been adopted by industries outside video gaming, such as film, automotive, architecture, engineering and construction.

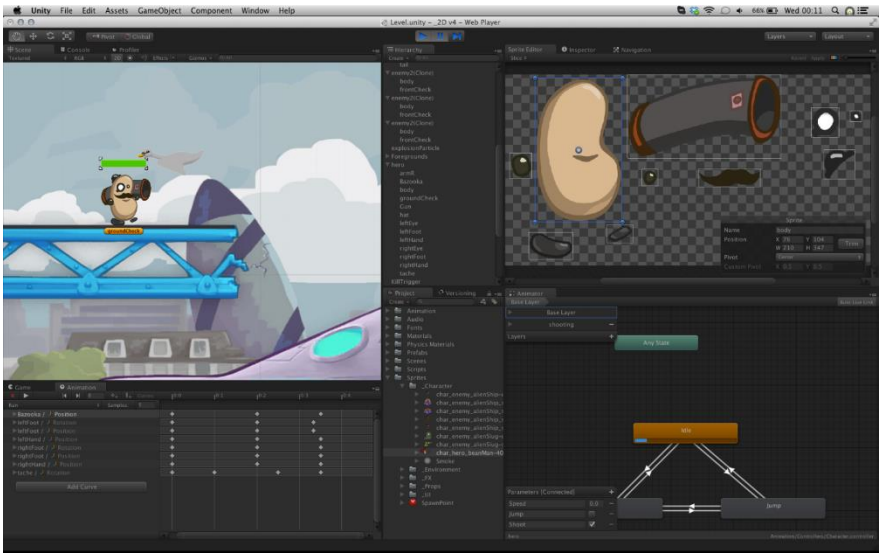


Figure 11: Unity engine editor

Arduino IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards.

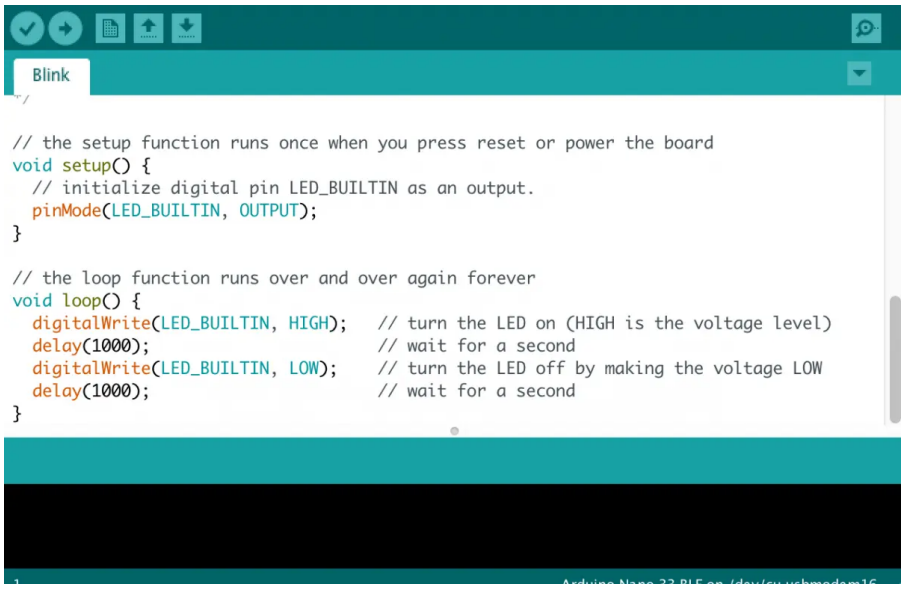


Figure 12: Arduino IDE editor

CHAPTER 3: METHODOLOGY

In order to be able to design this controller with its software, we worked on this methodology, which enabled us to reach a good result. The time was short and we could not implement everything, but we created this project by following these steps.

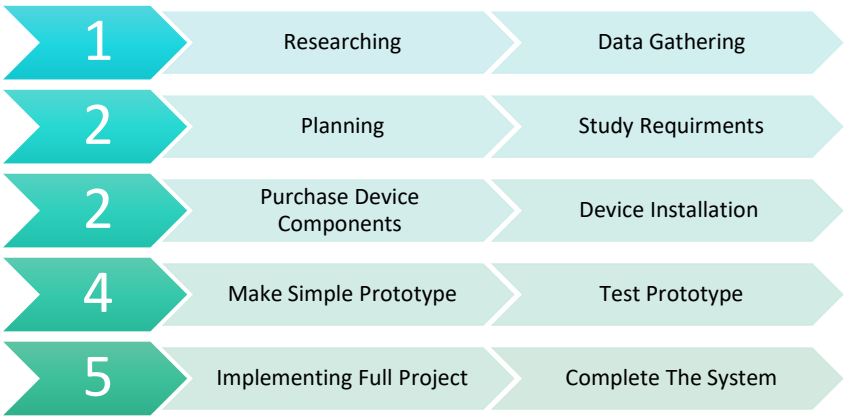


Figure 13: Multiple processes methodology

In the prototype stage, the project faced some challenges such that some parts did not function properly, as well as system instability. The initial development process was somewhat difficult, but by working on this methodology, we were able to arrive at the desired shape for the project.

CHAPTER 4: THEORY

Since we are dealing with two different environments, we have to find a way to make these systems understand each other. We found that the controller is a set of digital inputs such as buttons and joystick in addition to the motion and rotation sensor, the Arduino will read all these data in the form of numbers, and then we perform any calculations on them before collecting them next to each other and sending them.

The data will be sent next to some, separated by a specific letter, the receiving system will fully read this sent text, then it will separate the data by the method of determining the separator by the user.

Value	0,	0,	0,	0,	0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0
Shortcut	BL	BD	BU	BR	BO	JX	JY	GX	GY	GZ	AX	AY	AZ

Table 1: Simplified table of controller data

Shortcut	**Full Name**	**Datatype**
BL:	Button Left	int
BD:	Button Down	int
BU:	Button Up	int
BR:	Button Right	int
BO:	Button Options	int
JX:	Joystick X	float
JY:	Joystick Y	float
GX:	Gyroscope X	float
GY:	Gyroscope Y	float
GZ:	Gyroscope Z	float
AX:	Accelerometer X	float
AY:	Accelerometer Y	float
AZ:	Accelerometer Z	float

Table 2: Detailed table of controller data

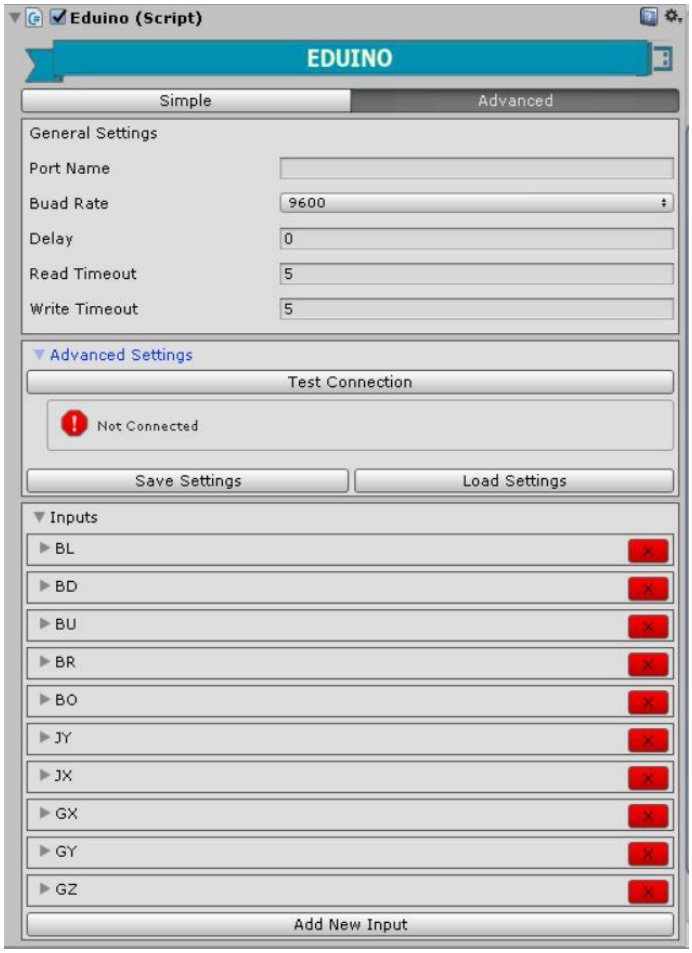


Figure 14: Eduino component in the inspector

The system, on the other hand, will separate the data through the separator, then it will return it to its original type or even change its type to what suits the user's requirements by specifying the pattern of receiving data from the specified interface.

Also, one of the basic things that the system will do is enable the user to search for available ports for use, find out which ones belong to the controller, and use them with adjusting the transmission speed settings and other things.

In the event that the user disconnects the controller from the game in one way or another, the system will enter the continuous search mode - which must work in the Multi-threading mode - to search for a controller connected to the device in order to be linked with the system.

One of the features of this system is that it allows the user to test the connection to the controller before playing the game, this will help the user to avoid problems with the connection to the controller. From the advanced options, the user can test the connection and search for available ports for use. He can also save and load the settings he has set so that they are not lost.

Finally, we named the system Eduino an acronym for Easy Arduino, meaning that it will enable you to easily connect with the Arduino-based controllers with your game.

CHAPTER 5: SYSTEM ANALYSIS

REQUIRMENTS

Design Requirements

- Giving the ability to the user to transfer any data he wants between Unity and Arduino using the API.
- Enable the user to design any controller he wants within the system.
- Make the API simple to the user.
- Create an editor that enables the user to read and use the inputs in his game directly.

Functions Requirement

- The system should use multi-threading to avoid a performance issue.
- A library provides functions for reading and writing data that is simple to use.
- A system that allows no contact with the controller to be lost.
- The system must be extensible.

Cost Requirement

Arduino & Components

Component	**Function**	**Quantity**	**Cost**	**Total Cost**
Arduino Nano	Microcontroller (the brain of all components)	1	15$	15$
Joystick	Horizontal & Vertical analog	1	2$	2$
Push Buttons	Purpose using	5	0.2$	1$
Resistors 10KΩ	For the buttons	5	0.2$	1$
MPU6050	Motion and rotation sensor	1	7$	7$
Wires	Connecting Components with the Arduino	~30	2$	2$
				28$

Table 3: Arduino and components cost

Computer (Minimum requirements)

Component	**Product**	**Cost**	**Total Cost**
CPU	Intel i5 8th	~150$	150$
GPU	Nvidia GTX1050 2GB	~100$	100$
Case	N/A	~40$	40$
Memory	DDR3 6GB	~80$	80$
Storage	500GB	~100$	100$
Monitor	N/A	~100$	100$
Other	N/A	~30$	30$
			600$

Table 4: Computer and components cost

UML

Flow Chart Diagram

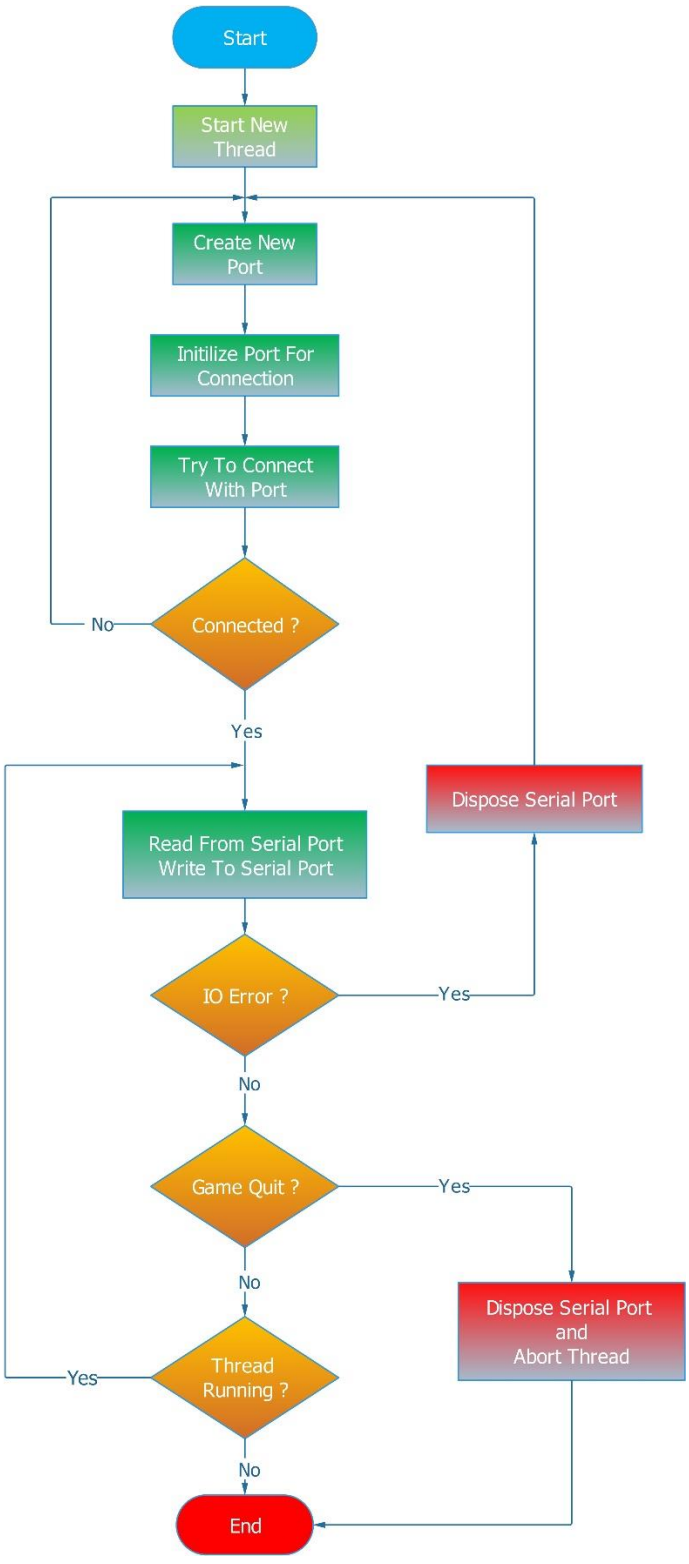


Figure 15: Flowchart

Data Flow Diagram

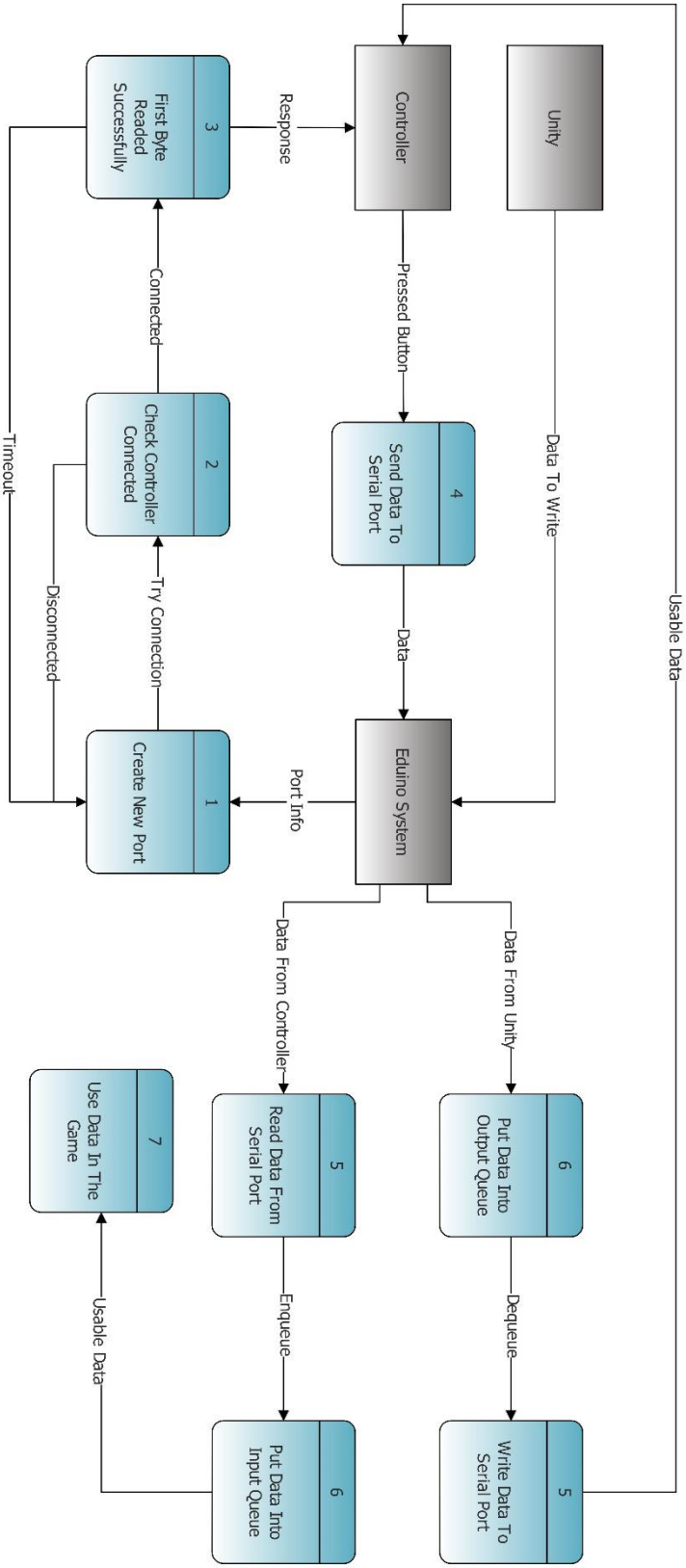


Figure 16: Data flow diagram (DFD)

State Machine Diagram

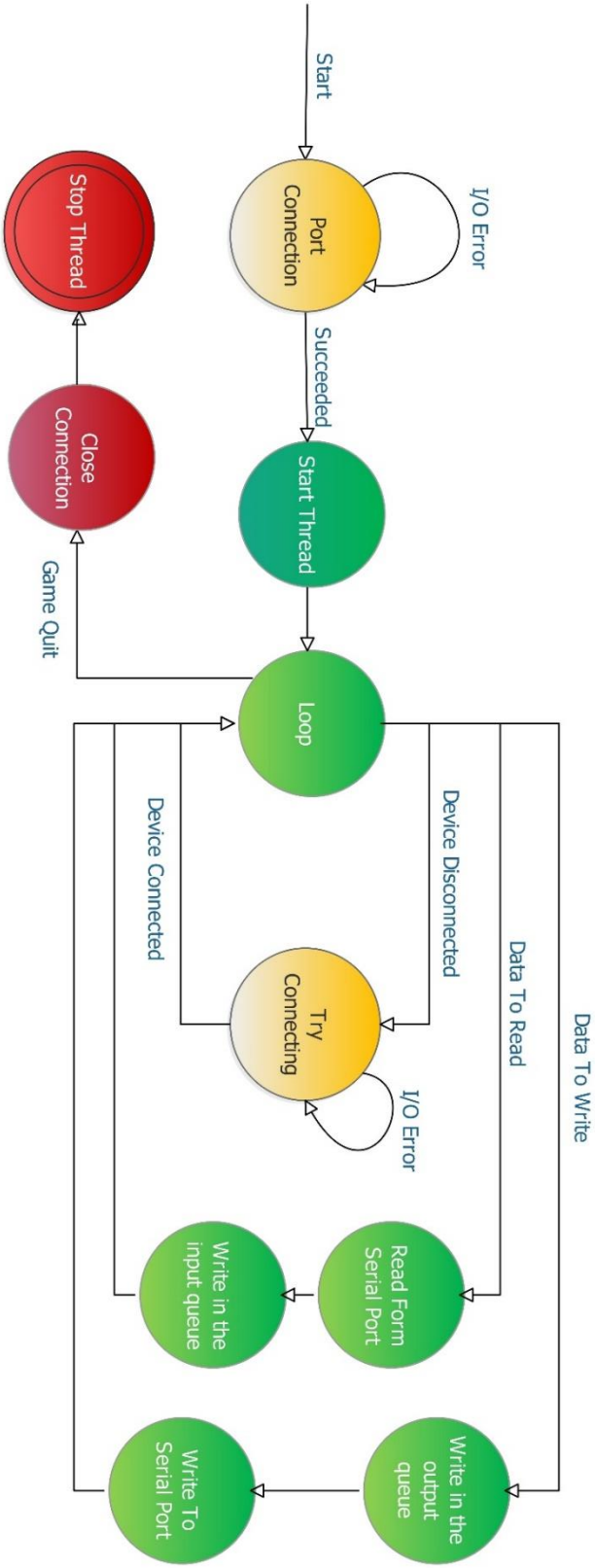


Figure 17: Finite State Machine (FSM)

CHAPTER 6: IMPLEMENTATION

CONTROLLER WIRING

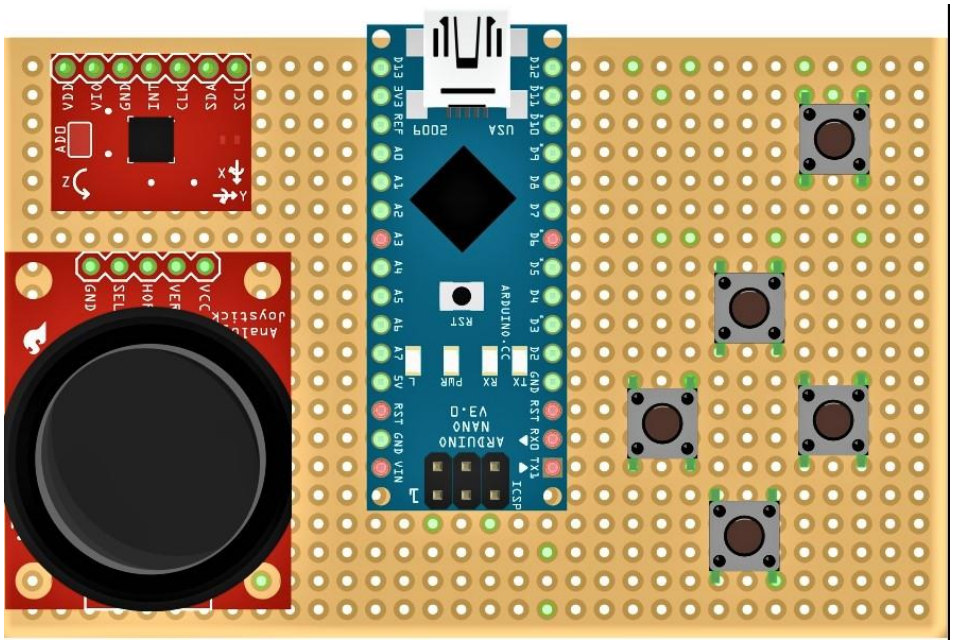


Figure 18: Front-end Controller Design

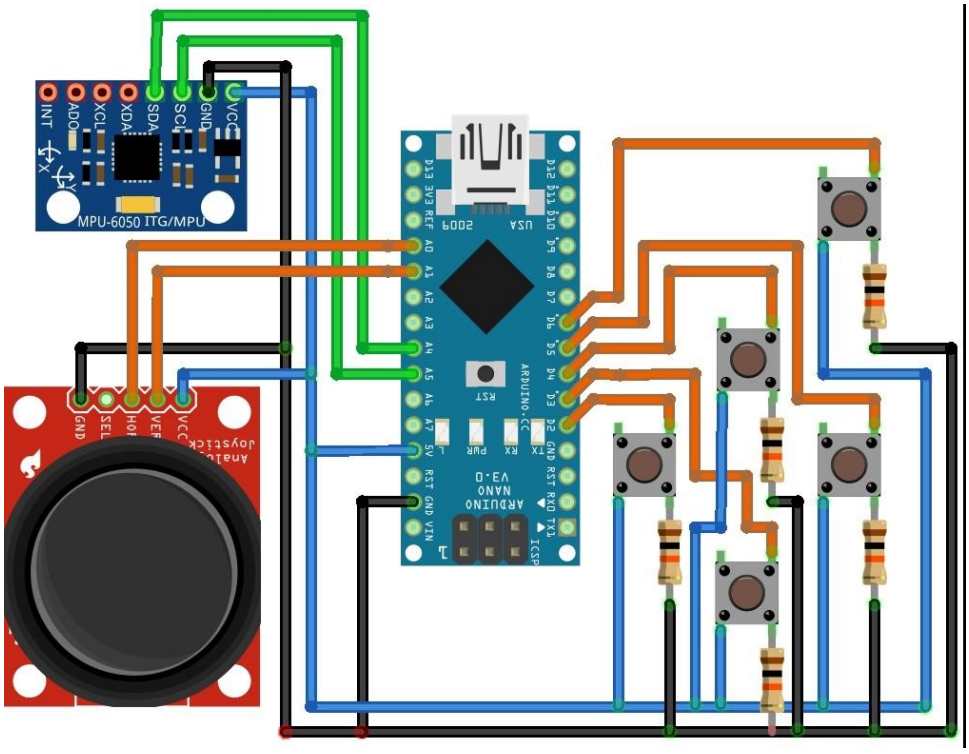


Figure 19: Back-end Controller wiring

ARDUINO CODE

```
#include <MPU6050_tockn.h>
#include <Wire.h>
#include <string.h>

MPU6050 mpu6050(Wire);

const int bl = 2, bd = 3, bu = 4, br = 5, bo = 6,jx = 14,jy = 15;

void setup() {

  Serial.begin(9600);
  pinMode(br, OUTPUT);
  pinMode(bd, OUTPUT);
  pinMode(bu, OUTPUT);
  pinMode(br, OUTPUT);
  pinMode(bo, OUTPUT);
  pinMode(jx, OUTPUT);
  pinMode(jy, OUTPUT);
  Wire.begin();
  mpu6050.begin();
  mpu6050.calcGyroOffsets(true);
}

void loop() {
  //BL, BD, BU, BR, BO, JX, JY, GX, GY, GZ

  Serial.print("Y");
  Serial.print(digitalRead(bl));
  Serial.print(",");
  Serial.print(digitalRead(bd));
  Serial.print(",");
  Serial.print(digitalRead(bu));
  Serial.print(",");
  Serial.print(digitalRead(br));
  Serial.print(",");
  Serial.print(digitalRead(bo));
  Serial.print(",");

  float x = analogRead(jx);
  float y = analogRead(jy);
  x = x / 1023 * 2 - 1; //map jx between -1 and 1
  y = y / 1023 * 2 - 1; //map jy between -1 and 1

  if(abs(x) < 0.05) //threshold
    x = 0;
  if(abs(y) < 0.05) //threshold
    y = 0;

  Serial.print(x);
  Serial.print(",");
  Serial.print(y);

  mpu6050.update();
  Serial.print(",");
  float gx = mpu6050.getAngleX() / 90; //map gx between -1 and 1
  float gy = mpu6050.getAngleY() / 90; //map gy between -1 and 1
  float gz = mpu6050.getAngleZ() / 90; //map gz between -1 and 1
  Serial.print(gy);
  Serial.print(",");
  Serial.print(gz);
  Serial.print(",");
  Serial.println(gx);
```

UNITY CODE

EduinoThread Code

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO.Ports;
using System.IO;
using UnityEngine.Events;
using System.Linq;
using System.Threading;


public abstract class EduinoThread {

    protected string portName;
    protected int delay;
    protected int readTimeout;
    protected int writeTimeout;
    public SafeSerialPort port{ get; protected set;}
    protected bool isRunning = true;
    protected int baudRateValue;
    private BaudRate baudRate;
    public SerialPortConnectionState ConnectionState{ get; private set;}

    protected Thread portThread;

    private Queue inputQueue, outputQueue;

    public EduinoThread(string portName, BaudRate buadRate, int delay, int readTimeout = 100, int writeTimeout
 = 100)
    {
        this.portName = portName;
        this.delay = delay;
        this.readTimeout = readTimeout;
        this.writeTimeout = writeTimeout;
        this.baudRateValue = int.Parse(buadRate.ToString ().Replace("m_", ""));
        this.baudRate = baudRate;

        inputQueue = Queue.Synchronized (new Queue ());
        outputQueue = Queue.Synchronized (new Queue ());
        SetState (SerialPortConnectionState.Disconnected);
    }

    public static string[] GetAvailablePorts()
    {
        return SafeSerialPort.GetPortNames ();
    }

    public SerialPort GetSerialPort()
    {
        return port;
    }

    public void Start()
    {
        lock(this)
            isRunning = true;

        portThread = new Thread (new ThreadStart(RunThread));
        portThread.Name = "Eduino Serial Port Thread";
        portThread.Start ();
    }


    public void Stop()
    {
        lock(this)
            isRunning = false;
    }
```

```
void RunThread()
{
    SetState (SerialPortConnectionState.Disconnected);

    while(IsRunning())
    {
        try {
            if(ConnectionState == SerialPortConnectionState.Disconnected)
                InitilizeConnection ();

            SetState (SerialPortConnectionState.Connected);

            if(ConnectionState == SerialPortConnectionState.Connected)
            {
                if(!port.IsOpen)
                    port.Open();

                if (port.IsOpen)
                {
                    ReadWrite();
                    OnUpdate();
                }else
                {
                    SetState (SerialPortConnectionState.Disconnected);
                }
            }

        } catch (IOException ex) {
            port.Dispose();
            OnConnectionFail();
            SetState (SerialPortConnectionState.Disconnected);
        }

        Debug.Log ("Port Connection : " + port.IsOpen);

        Thread.Sleep(delay);
    }

    SetState(SerialPortConnectionState.Stopped);
    OnThreadStop ();
}

public object ReadMessage()
{
    if (inputQueue.Count == 0 || inputQueue == null)
        return null;

    return (object)inputQueue.Dequeue ();
}

public void WriteMessage(string message)
{
    outputQueue.Enqueue (message);
}

void SetState(SerialPortConnectionState state)
{
    lock (this)
        ConnectionState = state;
}

void InitilizeConnection()
{
    while (ConnectionState == SerialPortConnectionState.Disconnected) {
        try {
            SetState(SerialPortConnectionState.Disconnected);
            OnTryConnection();

            string[] ports = GetAvailablePorts ();
            bool portFound = false;
            for (int i = 0; i < ports.Length; i++) {
```

```
                portName = ports[i];
                port = new SafeSerialPort (portName, baudRateValue);
                port.Open();
                port.ReadByte();
                OnConnected();
                portFound = true;
                break;
                Thread.Sleep(1000);
            }

            if(portFound)
                break;

            Thread.Sleep(1000);

        } catch (System.Exception ex) {
            SetState(SerialPortConnectionState.Disconnected);
            port.Dispose ();
            if(port.IsOpen)
                port.Close();
        }

        Thread.Sleep(1000);
    }

    Debug.Log(ConnectionState);

}

public static bool TestConnection(string _portName, BaudRate _buadRate)
{
    int testBaudRateValue = int.Parse(_buadRate.ToString ().Replace("m_", ""));
    SafeSerialPort testPort = new SafeSerialPort (_portName, testBaudRateValue);

    try {
        if(!testPort.IsOpen)
            testPort.Open();
    } catch (System.Exception ex) {
        Debug.Log ("error");
        testPort.Close ();
    }

    return testPort.IsOpen;
}

public static SafeSerialPort TryConnecteWithPort()
{
    string[] ports = GetAvailablePorts ();

    for (int i = 0; i < ports.Length; i++) {
        bool accepted = TestConnection (ports [i], BaudRate.m_9600);

        if (accepted) {
            int testBaudRateValue = int.Parse(BaudRate.m_9600.ToString ().Replace("m_", ""));
            return new SafeSerialPort (ports [i], testBaudRateValue);
        }
        Thread.Sleep (300);
    }

    return null;
}

public bool IsRunning()
{
    lock(this)
        return isRunning;
}



public void Disconnect()
{
```

```
        if (!IsRunning ())
            return;

        Stop();

        if(port != null)
        {
            if(port.IsOpen)
                port.Close();
        }

        portThread.Join ();
        Debug.Log (portThread.ThreadState);

        OnDisconnected ();
    }

    void ReadWrite()
    {
        string readedString = port.ReadLine ();

        if (inputQueue.Count > 5) {
            inputQueue.Dequeue ();
        }

        inputQueue.Enqueue (readedString);


        if (outputQueue.Count > 0) {
            port.WriteLine ((string)outputQueue.Dequeue ());
        }
    }

    protected virtual void OnTryConnection()
    {
        Debug.Log ("Try Connection...");
    }

    protected virtual void OnConnected()
    {
        Debug.Log ("Connected Successfully");
    }

    protected virtual void OnUpdate()
    {
    }

    protected virtual void OnConnectionFail()
    {
        Debug.Log ("Connection Fails");
    }

    protected virtual void OnThreadStop()
    {
        Debug.Log ("Thread has been stopped");
    }

    protected virtual void OnDisconnected()
    {
        Debug.Log ("Disconnected");
    }
}
```

EduinoSerialPort Code

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

public class EduinoSerialPort : EduinoThread {

    public event System.Action onUpdate;
    public event System.Action onConnected;
    public event System.Action onConnectionFail;
    public event System.Action onThreadStop;
    public event System.Action onTryConnection;

    public EduinoSerialPort(string portName, BaudRate buadRate, int delay, int readTimeout = 100, int writeTim
eout = 100)  : base(portName, buadRate, delay, readTimeout = 100, writeTimeout = 100)
    {

    }

    protected override void OnUpdate ()
    {
        if (onUpdate != null)
            onUpdate.Invoke ();
    }

    protected override void OnConnected ()
    {
        if (onConnected != null)
            onConnected.Invoke ();
    }

    protected override void OnConnectionFail ()
    {
        if (onConnectionFail != null)
            onConnectionFail.Invoke ();
    }

    protected override void OnThreadStop ()
    {
        if (onThreadStop != null)
            onThreadStop.Invoke ();
    }

    protected override void OnTryConnection ()
    {
        if (onTryConnection != null)
            onTryConnection.Invoke ();
    }

    protected override void OnDisconnected ()
    {
        onUpdate = null;
        onConnected = null;
        onConnectionFail = null;
        onThreadStop = null;
        onTryConnection = null;
    }
}
```

Eduino Code

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

public class Eduino : EduinoBase {

    public static Eduino Instance{ get; private set;}
    public List<EduinoDataInput> inputs;
    private string lastMessage = "";
    private string readedMessaged;

    [SerializeField] int settingsIndex = 0;

    void Awake()
    {
        if (Instance != null && Instance != this)
            Destroy (gameObject);

        Instance = this;
    }

    protected override void OnEnable ()
    {
        base.OnEnable ();

        serialPort.onUpdate += CalculateInputs;
    }

    public string ReadMessage()
    {
        string msg = readedMessaged;
        if (msg != null)
            lastMessage = msg;

        return lastMessage;
    }

    public void WriteMessage(string message)
    {
        serialPort.WriteMessage (message);
    }

    void CalculateInputs()
    {
        string data = (string)serialPort.ReadMessage ();
        readedMessaged = data;
        if (data != null) {
            string readedLine = data.ToString().Remove(0,1);
            string[] values = readedLine.Split(',');//BL, BD, BU, BR, BO, JY, JX,GX, GY, GZ
            int min = Mathf.Min(values.Length, inputs.Count);

            for (int i = 0; i < min; i++) {
                inputs[i].SetValue(values[i].ToString());
            }
        }
    }

    public EduinoDataInput FindInputByName(string inputName)
    {
        EduinoDataInput input = inputs.Find (t => t.name.Equals (inputName));
        if (input == null) {
            UnityEngine.Debug.LogError ("Input Name: " + inputName + "is not exists");
            return null;
        }
        return input;
    }



    public T GetValue<T>(string inputName)
```

```
    {
        EduinoDataInput input = inputs.Find (t => t.name.Equals (inputName));

        if (input == null) {
            return default(T);
        }

        return input.GetValue<T> ();
    }

    public bool GetTrigger(string inputName)
    {
        EduinoDataInput input = inputs.Find (t => t.name.Equals (inputName));;
        inputs.Find (t => t.name.Equals (inputName));

        if (input == null) {
            return false;
        }

        return input.GetTrigger();
    }
}
```

EduinoBase Code

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

public abstract class EduinoBase : MonoBehaviour {

    public string portName;
    public BaudRate buadRate;
    public int delay;
    public int readTimeout = 100;
    public int writeTimeout = 100;

    protected EduinoSerialPort serialPort;

    public SerialPortConnectionState ConnectionState{
        get{
            return serialPort.ConnectionState;
        }
    }

    public bool IsConnected{
        get{
            return serialPort.ConnectionState == SerialPortConnectionState.Connected;
        }
    }

    public bool IsDisconnected{
        get{
            return serialPort.ConnectionState == SerialPortConnectionState.Disconnected;
        }
    }

    protected virtual void OnEnable()
    {
        serialPort = new EduinoSerialPort (portName, buadRate, delay, readTimeout, writeTimeout);
        serialPort.Start ();

    }

    protected virtual void OnDisable()
    {
        print ("end");
        if(serialPort != null)
            serialPort.Disconnect ();
    }

    protected virtual void OnDestroy()
    {
        if(serialPort != null)
            serialPort.Disconnect ();
    }

    protected virtual void OnApplicationQuit()
    {
        if(serialPort != null)
            serialPort.Disconnect ();
    }
}
```

EduinoDataInput Code

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

[System.Serializable]
public class EduinoDataInput{
    public enum InputDataType{
        String,
        Boolean,
        Float,
        Int
    }

    [System.Serializable]
    public class InputOptions
    {
        public float triggerThreshold = 0;
        public bool flipValue = false;
    }

    public string Value { get; private set;}
    public string name;
    public InputDataType type;
    public InputOptions options;

    private bool isChanged = false;

    public void SetValue(string _value)
    {
        Value = _value;
    }

    public T GetValue<T>()
    {
        return (T)GetValue ();
    }

    private object GetValue()
    {
        switch (type) {
        case InputDataType.Boolean:
            bool btn = (Value == "1") ? true : false;
            return options.flipValue ? !btn : btn;

        case InputDataType.Float:

            float fres = 0;
            float.TryParse (Value, out fres);
            return options.flipValue ? -fres : fres;

        case InputDataType.Int:
            int ires = 0;
            int.TryParse (Value, out ires);
            return options.flipValue ? -ires : ires;

        case InputDataType.String:
            return Value.ToString ();
        }

        return Value;
    }
    public bool GetTrigger()
    {
        switch (type) {
        case InputDataType.Boolean:


            bool btn = false;
            if (!isChanged) {
                btn = (Value == "1") ? true : false;
```

```
                isChanged = true;
            }

            if (Value == "0")
                isChanged = false;

            return btn;

        case InputDataType.Float:

            bool fstate = false;
            float fres = 0;
            float.TryParse (Value, out fres);

            if (!isChanged) {
                if (Mathf.Abs(fres) >= options.triggerThreshold) {
                    isChanged = true;
                    fstate = true;
                }
            }

            if (Mathf.Abs(fres) < options.triggerThreshold)
                isChanged = false;

            return fstate;

        }

        return false;
    }
}
```

CHAPTER 7: RESULTS

EDUINO INTERFACE

The Eduino interface consists of 3 sections

General Settings

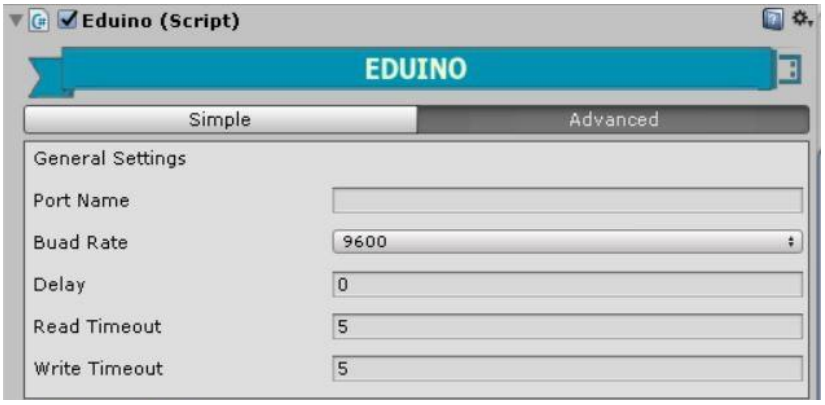


Figure 20: Eduino General Settings

Advanced Settings

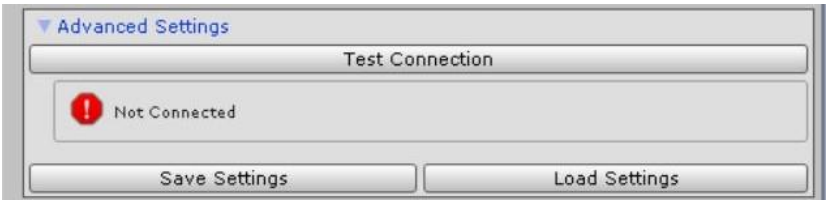


Figure 21: Eduino Advanced Settings

Inputs Settings

Figure 22: Eduino Inputs Settings

THE USING OF THE EDUINO SYSTEM

Example Code For Using Inputs:

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

public class TestEduinoSystem : MonoBehaviour {

    EduinoDataInput joystickX;
    EduinoDataInput leftButton;

    void Start () {

        joystickX = Eduino.Instance.FindInputByName ("JX");
        leftButton = Eduino.Instance.FindInputByName ("BL");
    }

    void Update () {

        print (joystickX.GetValue<float> ());
        print (joystickX.GetTrigger ()); //return true when this value pass the threshold
        print (leftButton.GetValue<bool> ());
    }
}
```

Example Code For Writing Message To Controller:

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

public class TestEduinoSystem : MonoBehaviour {

    void Start () {
        Eduino.Instance.WriteMessage ("Send To Controller");
    }
}
```

Example Code For Reading Message From Controller:

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

public class TestEduinoSystem : MonoBehaviour {

    void Start () {
        string message Eduino.Instance.ReadMessage ();
        print(message);
    }
}
```

THE FINAL VIEW OF THE PROJECT

The Game Controller:

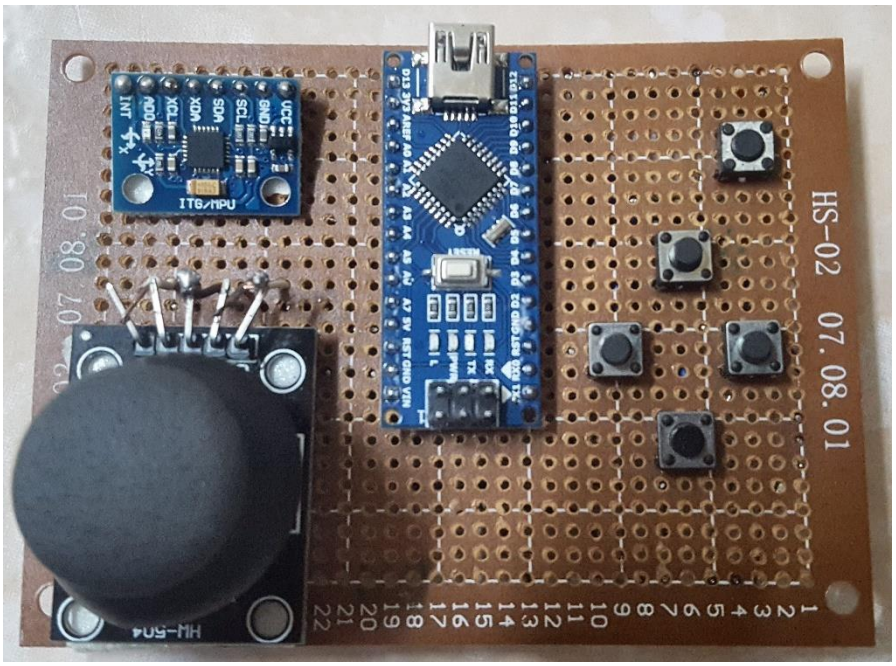


Figure 23: The final look of the controller

The Game Project and Eduino System Inside Unity Engine:

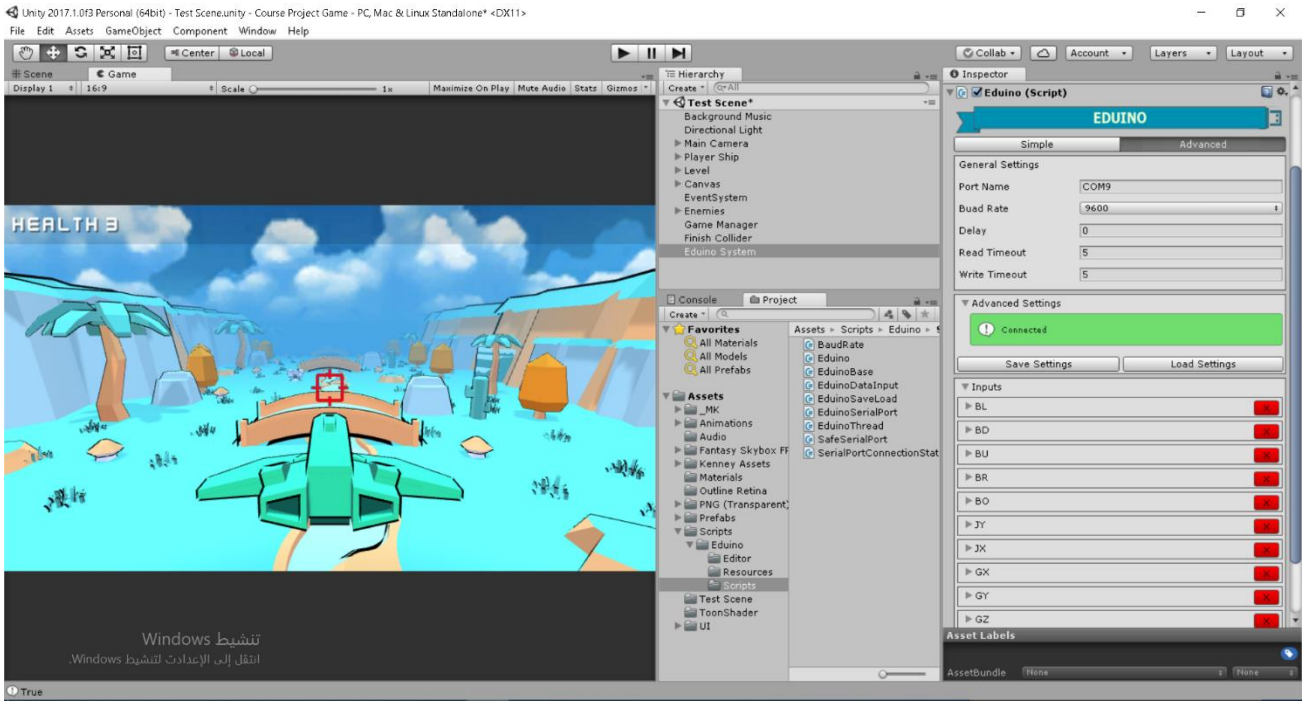


Figure 24: The example game inside unity engine

CHAPTER 8: CONCLUSION

In the end, we hope that we have presented our project to the fullest, and we hope that it will be of great value to you. We have barely scratched the surface of the controller design, and this is our first step and our first experience in designing a controller with its own integrated system.

We wished that we had enough time to design a suitable shape for this controller and perhaps reduce its size and get rid of unnecessary parts, but we did not have time to do this.

Finally, we hope that this type of system will be given enough attention, and we hope that future generations will pay attention to such topics as they are very useful and highly practical, and that they develop this system and bring it to a higher level.