**INTRODUCTION**

**A.1 ABSTRACT**

IOT Simulator is an easy to use, IOT Sensor/device simulator that quickly creates test environments made up of thousands of sensors and gateways, all on just one computer.

IOT Simulator supports many of the common IOT protocols.

It enables IOT platform and gateway vendors to improve product quality and significantly shorten their time-to-market without incurring large capital expense for creating test infrastructure. Both IPv4 and IPv6 sensors are supported and the simulator includes built-in support for lossy behaviour in constrained environments. It can also create scripted error scenarios on demand. Example scripts to work with popular platforms like Azure IOT, Amazon AWS, IBM Blue mix and others are available to facilitate quick setup.

**A.2 INTRODUCTION**

The project is mainly about enhancing the vision of simulating the IOT experiments through the help of Environment. It is worth because of the price & other factors like lack of IOT kits in College Labs, our application intends to solve this issue with introducing free & ready to learn app.

**Objective**:

* To create an environment using Unity
* To provide a platform for stable environment
* The software to provide IOT experience

**Benefits**:

The software is useful for variety of users. Since the software is useful for students they can simulate the basic working model of the algorithm. The universities, colleges can no longer need to specifically invest in the lab kits.

1. **LITERATURE SURVEY**

**B.1 EXISTING SYSTEM**

Currently there are no application which provide these simulation experience as a software. Our software provides a way of creating the environment towards a single standalone software.

**B.2 PROPOSED SYSTEM**

We established a conceptual theme that simulates the IOT for experience. The advanced features mean to create a way of learning and enhanced learning capacities for the new learners. Our system simulates how a component work and simulates on real time scenarios. This in term helps to learn the different sensors which can be attached to controller.

The proposed system discusses the blueprint system.

**B.3 FEASIBILITY STUDY**

**CPU :**

Total CPU: 0.08ms

Input Lighting: 0.03ms

Solve Tasks: 0.02ms

Dynamic Objects: 0.02ms

Time Between Updates: 4.96ms

**Memory**

Used Total: 186.5 MB Unity: 80.2 MB Mono: 11.2 MB GfxDriver: 11.5 MB Audio: 2.0 MB Video: 0 B Profiler: 81.5 MB

Reserved Total: 352.5 MB Unity: 224.2 MB Mono: 12.8 MB GfxDriver: 11.5 MB Audio: 2.0 MB Video: 0 B Profiler: 102.0 MB

Total System Memory Usage: 0.87 GB

1. **SYSTEM REQUIREMENTS**

**C.1 LANGUAGE USED**

**C#** (pronounced *see sharp*, like the musical note [C♯](https://en.wikipedia.org/wiki/C%E2%99%AF_(musical_note)), but written with the [number sign](https://en.wikipedia.org/wiki/Number_sign))is a general-purpose, [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language) encompassing [strong typing](https://en.wikipedia.org/wiki/Strong_typing), [lexically scoped](https://en.wikipedia.org/wiki/Lexically_scoped), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [declarative](https://en.wikipedia.org/wiki/Declarative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming), [generic](https://en.wikipedia.org/wiki/Generic_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) ([class](https://en.wikipedia.org/wiki/Class_(computer_science))-based), and [component-oriented](https://en.wikipedia.org/wiki/Component-based_software_engineering) programming disciplines.[[16]](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)#cite_note-ECMA-334-18) It was developed around 2000 by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) as part of its [.NET](https://en.wikipedia.org/wiki/.NET_Framework) initiative, and later approved as an [international standard](https://en.wikipedia.org/wiki/International_standard) by [Ecma](https://en.wikipedia.org/wiki/Ecma_International" \o "Ecma International) (ECMA-334) and [ISO](https://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO/IEC 23270:2018). [Mono](https://en.wikipedia.org/wiki/Mono_(software)) is the name of the free and open-source project to develop a compiler and runtime for the language. C# is one of the programming languages designed for the [Common Language Infrastructure](https://en.wikipedia.org/wiki/Common_Language_Infrastructure) (CLI).

C# was designed by [Anders Hejlsberg](https://en.wikipedia.org/wiki/Anders_Hejlsberg), and its development team is currently led by [Mads Torgersen](https://en.wikipedia.org/w/index.php?title=Mads_Torgersen&action=edit&redlink=1" \o "Mads Torgersen (page does not exist)). The most recent version is 8.0, which was released in 2019 alongside [Visual Studio 2019](https://en.wikipedia.org/wiki/Visual_Studio_2019) version 16.3.

**C.2 TOOLS AND TECHNOLOGIES USED**

**Unity** is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [game engine](https://en.wikipedia.org/wiki/Game_engine) developed by [Unity Technologies](https://en.wikipedia.org/wiki/Unity_Technologies), first announced and released in June 2005 at [Apple Inc.](https://en.wikipedia.org/wiki/Apple_Inc.)'s [Worldwide Developers Conference](https://en.wikipedia.org/wiki/Apple_Worldwide_Developers_Conference) as a [Mac OS X](https://en.wikipedia.org/wiki/MacOS)-exclusive game engine. As of 2018, the engine had been extended to support more than 25 platforms. The engine can be used to create [three-dimensional](https://en.wikipedia.org/wiki/Three-dimensional_space), [two-dimensional](https://en.wikipedia.org/wiki/Two-dimensional_space), virtual reality, and augmented reality games, as well as [simulations](https://en.wikipedia.org/wiki/Computer_simulation) and other experiences.[[4]](https://en.wikipedia.org/wiki/Unity_(game_engine)#cite_note-Easier-4)[[5]](https://en.wikipedia.org/wiki/Unity_(game_engine)#cite_note-5) The engine has been adopted by industries outside video gaming, such as [film](https://en.wikipedia.org/wiki/Film_industry), [automotive](https://en.wikipedia.org/wiki/Automotive_industry), [architecture](https://en.wikipedia.org/wiki/Architecture), [engineering](https://en.wikipedia.org/wiki/Engineering) and [construction](https://en.wikipedia.org/wiki/Construction).

Several major versions of Unity have been released since its launch. The latest stable version, 2019.2.13, was released in November 2019

**D.SOFTWARE REQUIREMENTS SPECTIFICATION**

**D1. FUNCTIONAL REQUIREMENTS**

Introduction to the components

In Introduction module, it contains all the components & associated description for better understanding hands on sessions. It contains the preview feature for example LED light on/off.

Demo project Module

In Demo project module, one sample project is created to simulate the established connections.

Simulation Module

In Simulation module, the components and other sensors are provided to create a project. It simulates a stable environment for the project.

**D2. NON-FUNCTIONAL REQUIREMENTS**

* Performance : The system performance should be high enough to handle all the sensors and it’s activity.
* Reliability : The system is consistently good in quality and performance.
* Speed : Our system gives quick response to the user requests.
* Ease of use : The user interface is user friendly that a user with some knowledge of using electronic devices can user the application.

**E. SYSTEM DESIGN**

**E.1 CLASS DIAGRAM**

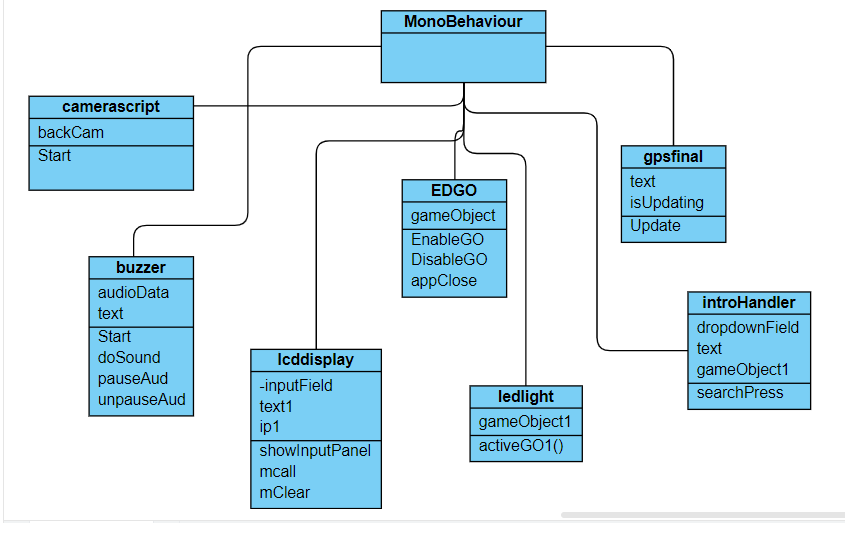


Figure : E.1 : Class Diagram

**E.2 ACTIVITY DIAGRAM**

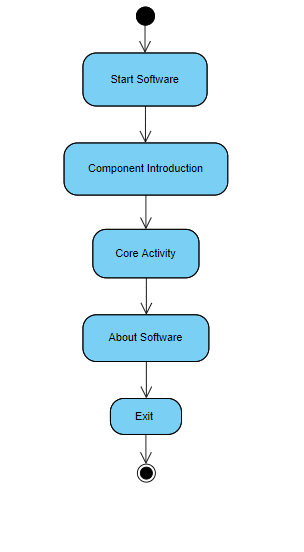


Figure : E.2 : Activity Diagram

**E.3 USECASE DIAGRAM**

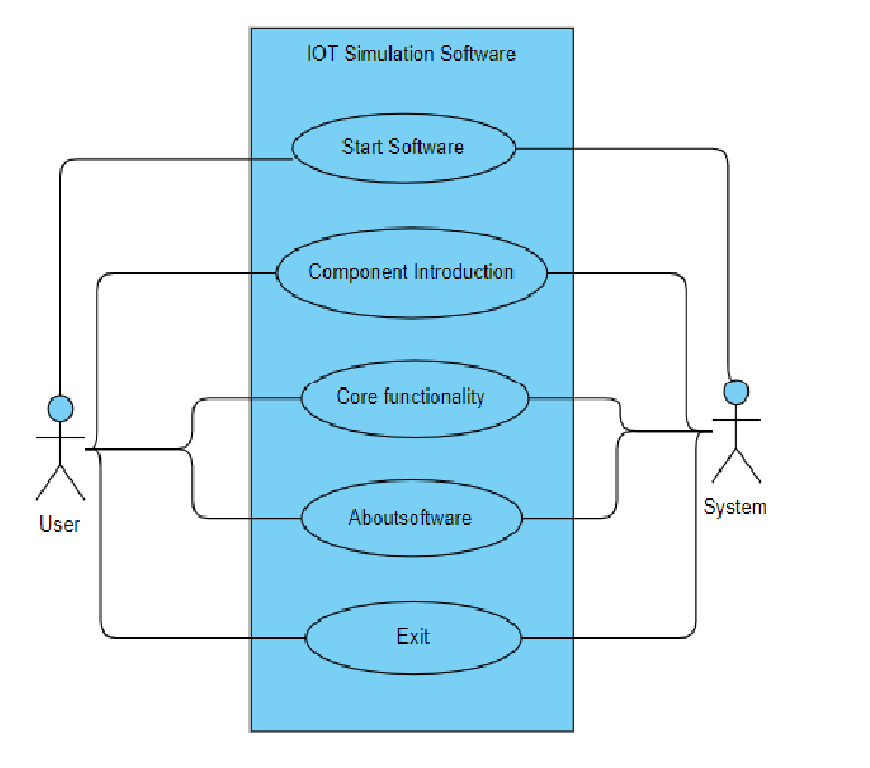


Figure : E.3 : Use case Diagram

**F. IMPLEMENTATION**

**F.1 SOURCE CODE**

**change.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class change : MonoBehaviour

{

// Start is called before the first frame update

public void changeScene(string sname)

{

SceneManager.LoadScene(sname);

Debug.Log("SceneLoaded" + sname);

}

}

**camerascript.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class camerascript : MonoBehaviour

{

static WebCamTexture backCam;

void Start()

{

if (backCam == null)

backCam = new WebCamTexture();

GetComponent<Renderer>().material.mainTexture = backCam;

if (!backCam.isPlaying)

backCam.Play();

}

}

**Buzzer.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class buzzer : MonoBehaviour

{

public AudioSource audioData;

public Text text;

void Start()

{

audioData = GetComponent<AudioSource>();

doSound();

}

public void doSound(){

audioData.Play(0);

Debug.Log("started");

text.text = "Buzzer creates sound \n use buttons to pause or continue";

}

public void pauseAud()

{

audioData.Pause();

Debug.Log("Pause: " + audioData.time);

}

public void unpauseAud()

{

audioData.UnPause();

}

}

**EDGO.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EDGO : MonoBehaviour

{

//Enable Disable GameObject

public GameObject gameObject;

public void EnableGO() {

gameObject.SetActive(true);

}

public void DisableGO()

{

gameObject.SetActive(false);

}

public void appClose()

{

Application.Quit();

}

}

**introHandler.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class introHandler : MonoBehaviour

{

//var declarations

public Dropdown dropdownField;

public Text text;

public GameObject[] gameObject1;

public void searchPress()

{

Debug.Log(dropdownField.options[dropdownField.value].text);

text.text = "" + dropdownField.options[dropdownField.value].text;

if (text.text == "Buzzer")

{

text.text = "Buzzer : Micro Buzzer-5V-Black are used for making beeps, tones and alerts. To use, connect short pin to ground and the other pin to 5 voltage level.A passive buzzer (AKA magnetic transducer) can make different tones, but the devices that controls the buzzer has to provide it with an oscillating electronic signal at a desired frequency. The supplied frequency will determine the tone. Supplying just a fixed voltage will generate no sound, except perhaps a slight tick at the point when the power source is connected or disconnected from the buzzer.";

for (int i = 0; i < gameObject1.Length; i++)

{

gameObject1[i].SetActive(false);

}

gameObject1[0].SetActive(true);

}

if (text.text == "LCD Display")

{

text.text = " The LiquidCrystal() function sets the pins the Arduino uses to connect to the LCD. You can use any of the Arduino's digital pins to control the LCD. Just put the Arduino pin numbers inside the parentheses in this order: LiquidCrystal(RS, E, D4, D5, D6, D7). RS, E, D4, D5, D6, D7 are the LCD pins.";

for (int i = 0; i < gameObject1.Length; i++)

{

gameObject1[i].SetActive(false);

}

gameObject1[1].SetActive(true);

}

if (text.text == "Camera")

{

text.text = "This is an Arduino camera module, adopted the Surveillance cameras digital image processing chip-OV0706, specially designed for image acquisition and processing application, based on TTL communication interface, very convenient to connect with Arduino controller, able to read image ";

for (int i = 0; i < gameObject1.Length; i++)

{

gameObject1[i].SetActive(false);

}

gameObject1[2].SetActive(true);

}

if (text.text == "GPS")

{

text.text = "The Global Positioning System (GPS), originally NAVSTAR GPS,[1] is a satellite-based radionavigation system owned by the United States government and operated by the United States Air Force.[2] It is a global navigation satellite system (GNSS) that provides geolocation and time information to a GPS receiver anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites.[3] Obstacles such as mountains and buildings block the relatively weak GPS signals.";

for (int i = 0; i < gameObject1.Length; i++)

{

gameObject1[i].SetActive(false);

}

gameObject1[4].SetActive(true);

}

if (text.text == "LED")

{

text.text = "The light sensor is a passive devices that convert this “light energy” whether visible or in the infra-red parts of the spectrum into an electrical signal output. Light sensors are more commonly known as “Photoelectric Devices” or “Photo Sensors” because the convert light energy (photons) into electricity (electrons).";

for (int i = 0; i < gameObject1.Length; i++)

{

gameObject1[i].SetActive(false);

}

gameObject1[3].SetActive(true);

}

}

}

**Gpsfinal.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.Android;

public class gpsfinal : MonoBehaviour

{

public Text text;

public bool isUpdating;

void Update()

{

if (!isUpdating)

{

StartCoroutine(GetLocation());

isUpdating = !isUpdating;

}

IEnumerator GetLocation()

{

if (!Permission.HasUserAuthorizedPermission(Permission.FineLocation))

{

Permission.RequestUserPermission(Permission.FineLocation);

Permission.RequestUserPermission(Permission.CoarseLocation);

}

if (!Input.location.isEnabledByUser)

{

Debug.Log("User Has Not Enabled GPS");

text.text = "User Has Not Enabled GPS";

yield return new WaitForSeconds(3);

}

Input.location.Start();

int maxWait = 3;

while (Input.location.status == LocationServiceStatus.Initializing && maxWait > 0)

{

yield return new WaitForSeconds(1);

maxWait--;

}

if (maxWait <= 0)

{

Debug.Log("Timed Out!");

text.text = "Timed Out";

yield break;

}

if (Input.location.status == LocationServiceStatus.Failed)

{

Debug.Log("Unable to determine device location!");

text.text = "Unable to determine device location!";

yield break;

}

else

{

text.text = "Lat : " + Input.location.lastData.latitude + "Long : " + Input.location.lastData.longitude;

}

isUpdating = !isUpdating;

Input.location.Stop();

}

}

}

**Ledlight.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ledlight : MonoBehaviour

{

public GameObject[] gameObject1;

public void activeGO1(int n)

{

for (int i = 0; i < gameObject1.Length; i++)

{

gameObject1[i].SetActive(false);

}

gameObject1[n].SetActive(true);

}

}

**Lcddisplay.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class lcddisplay : MonoBehaviour

{

public InputField inputField;

public Text text1;

private string ip1;

public void showInputPanel()

{ ip1 = inputField.text;

}

public void mcall()

{

if (inputField.text.Length <= 12)

text1.text = "" + inputField.text;

else

text1.text = "Text Length > 12 can't display the text";

}

public void mClear() {

inputField.text = "";

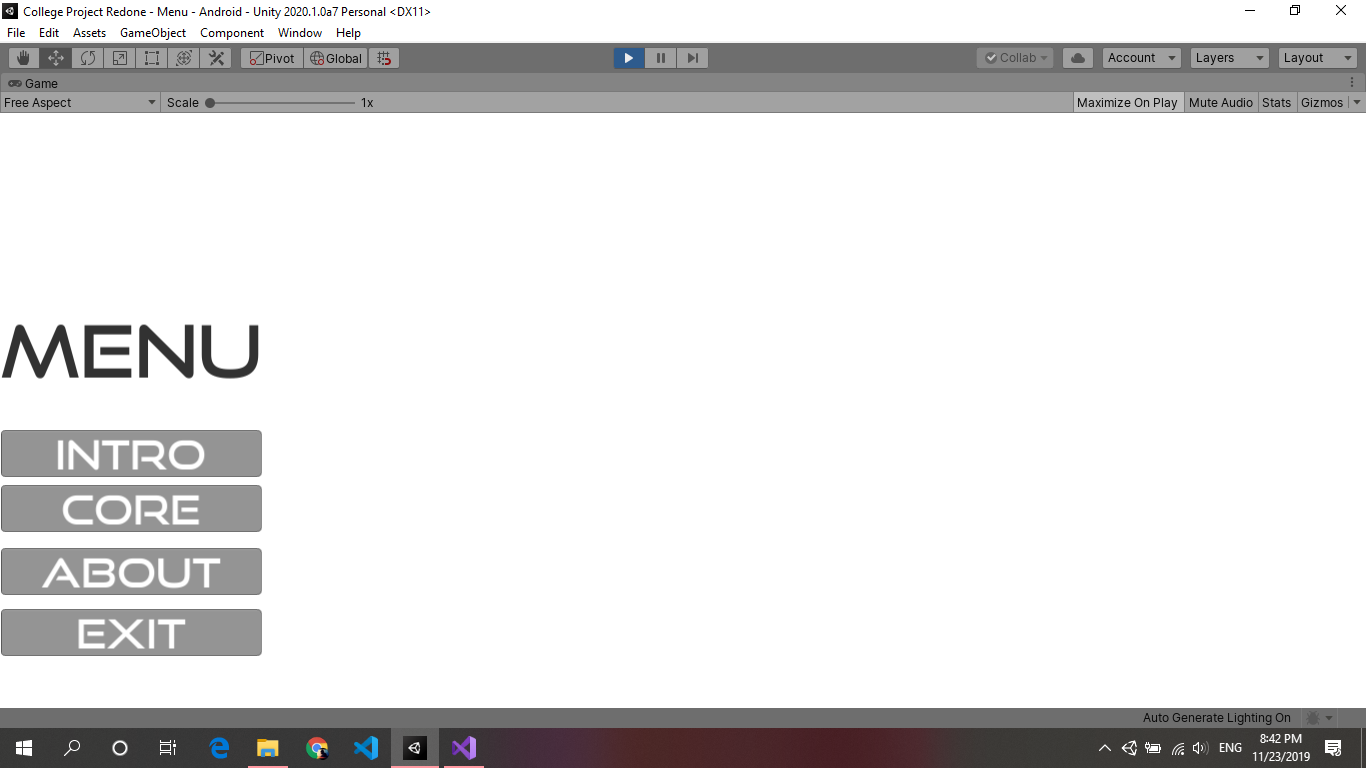
text1.text = "";

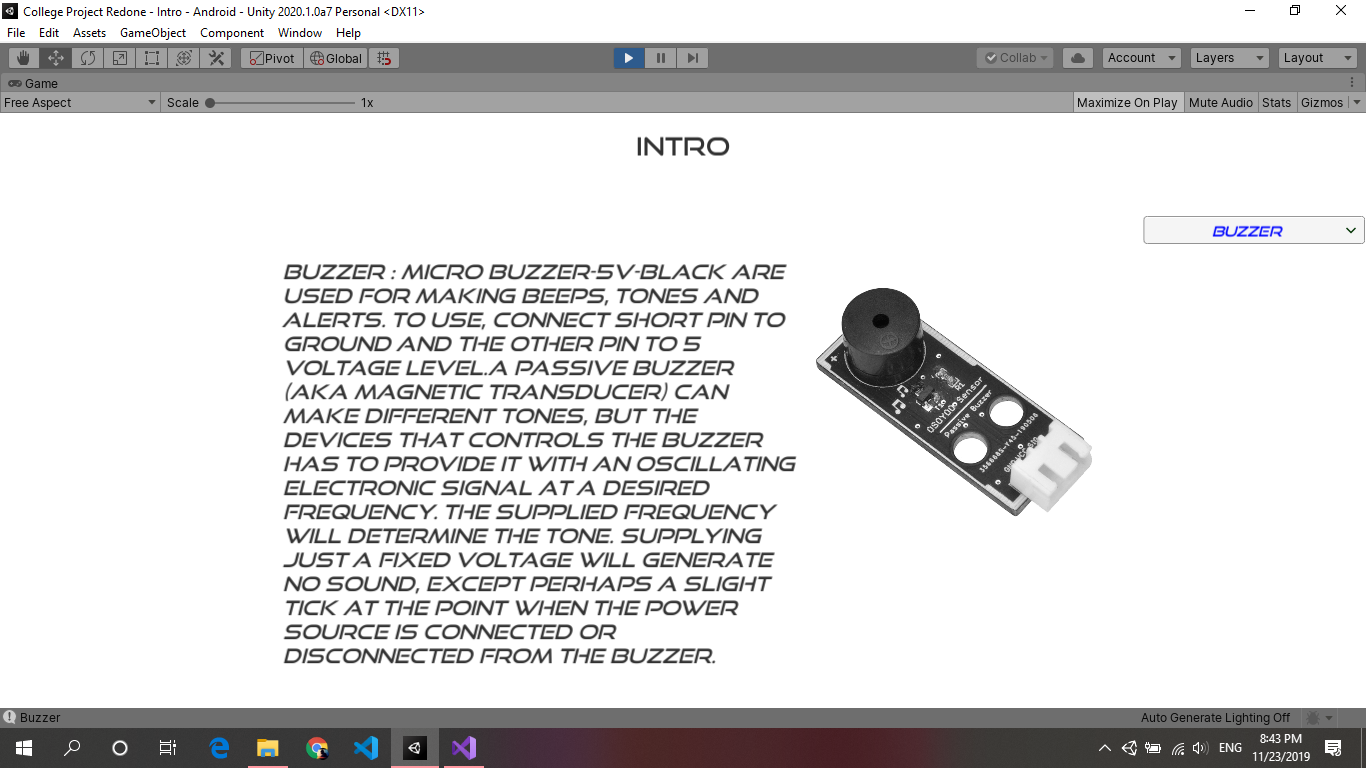
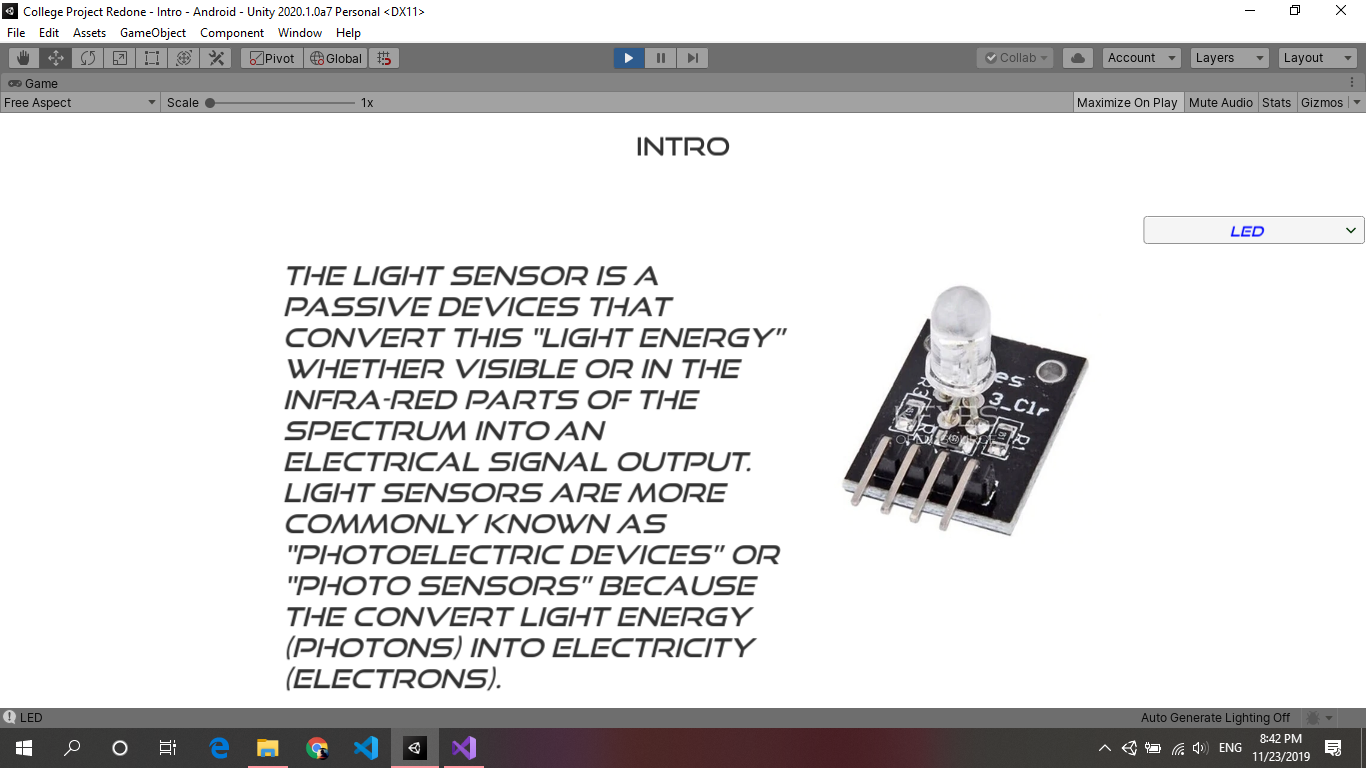
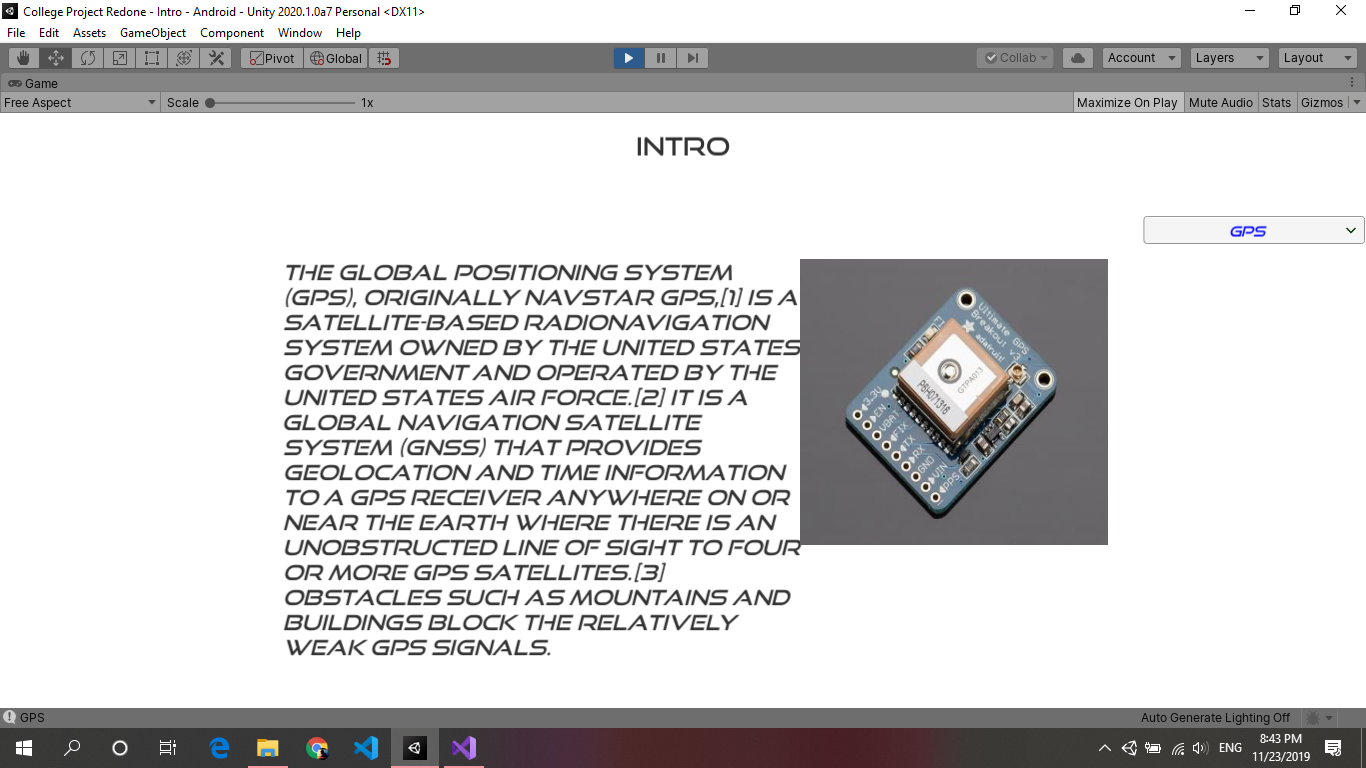
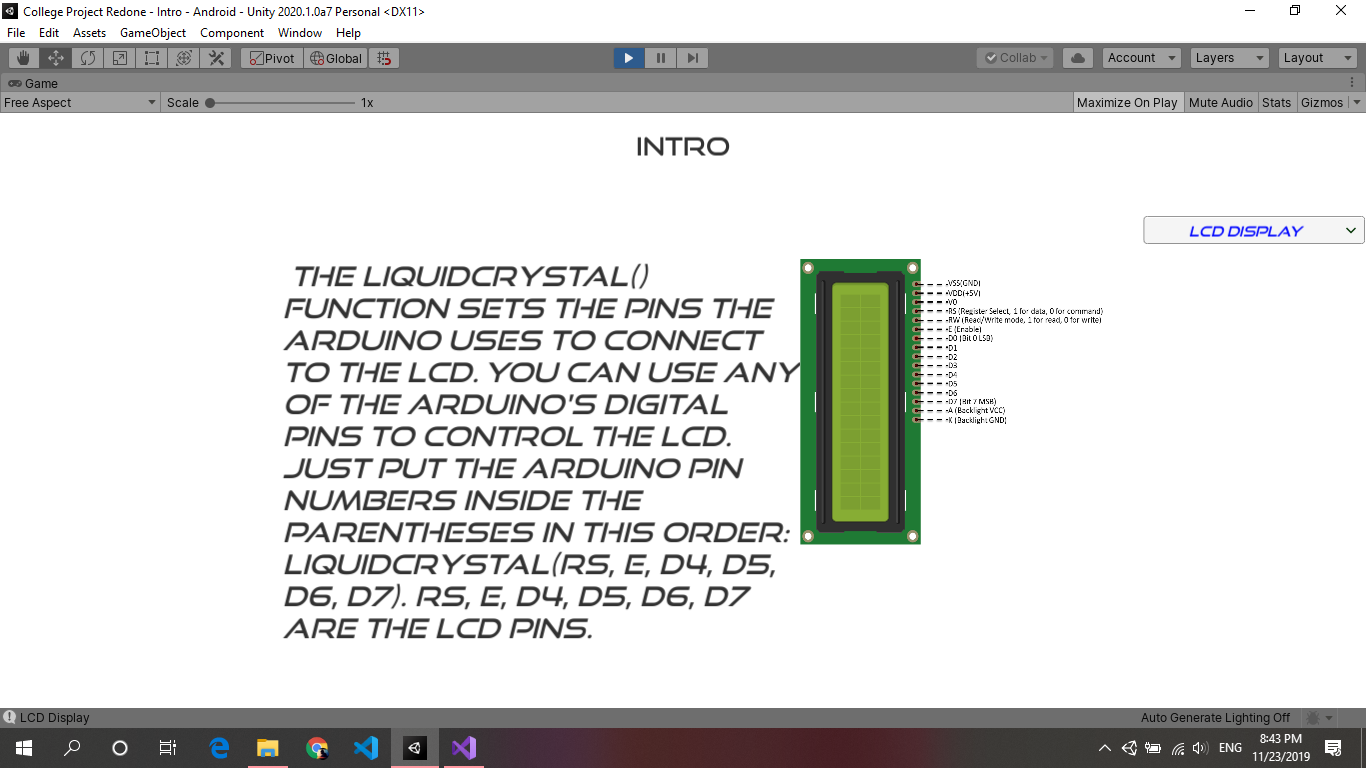
}

}

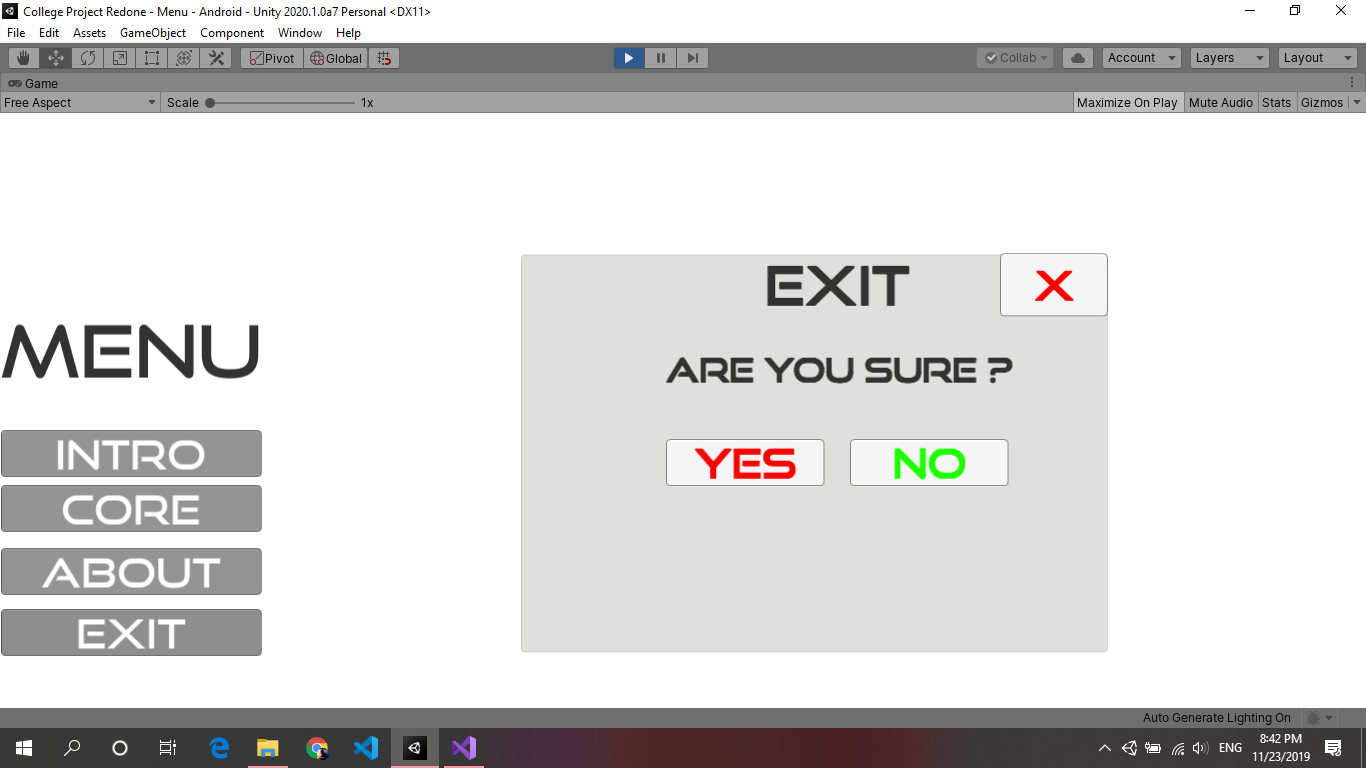
**F.2 SCREEN SHOTS**

**1.MENU**

****

**2.INTRO**

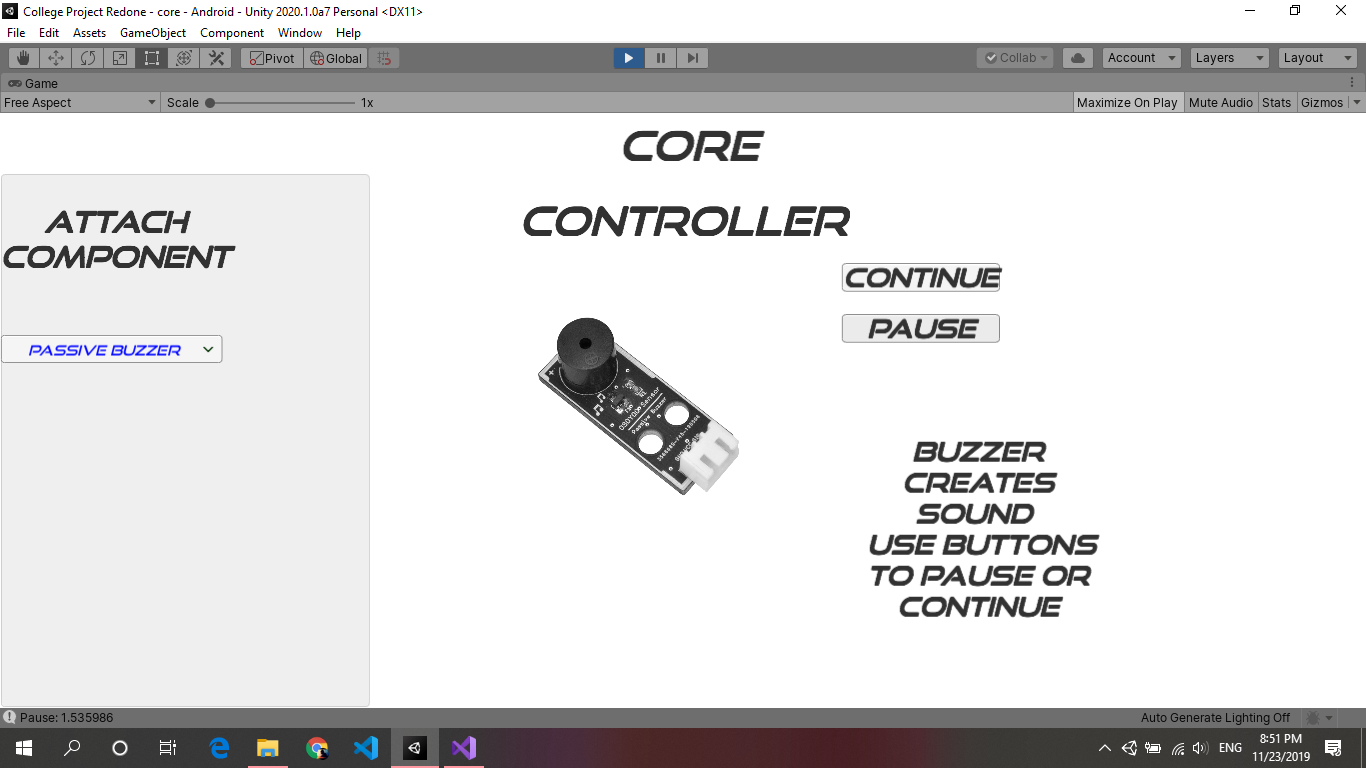
**3.EXIT**

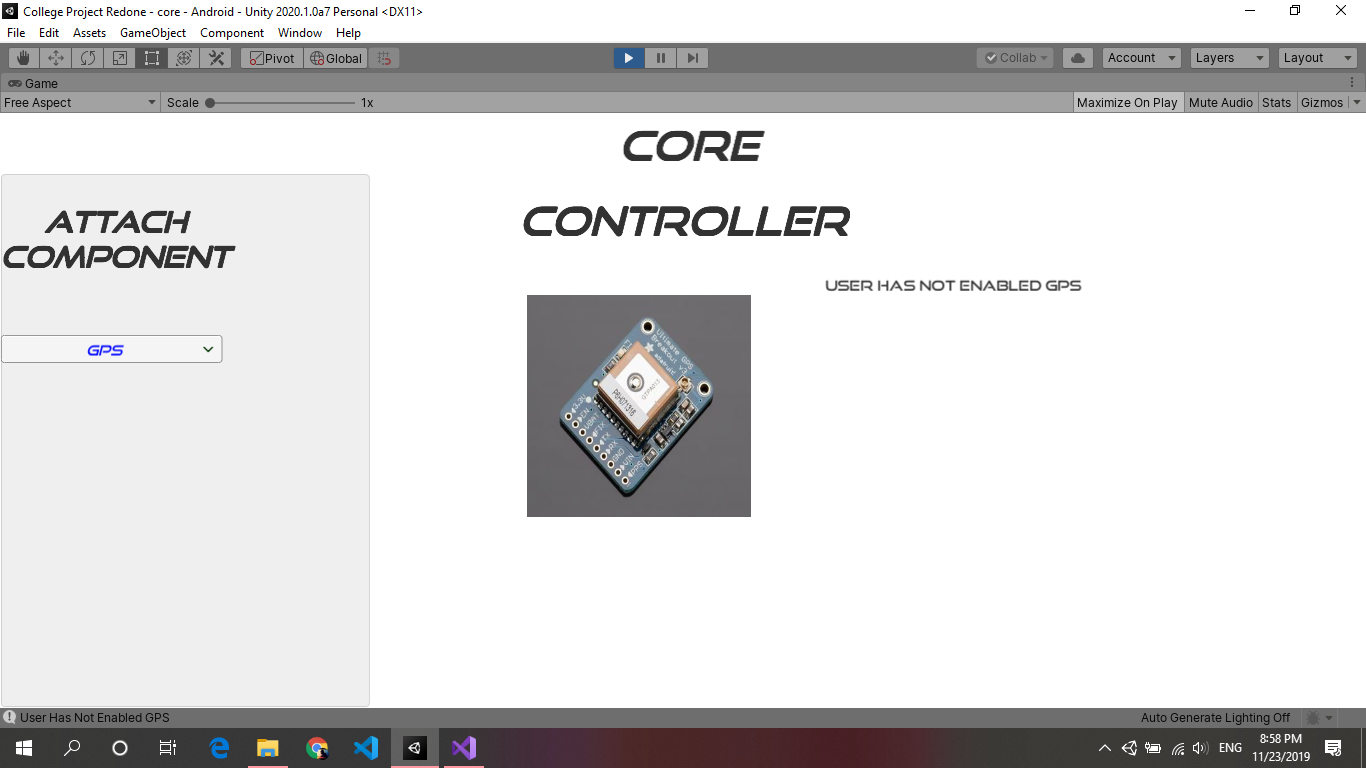
****

**4.ABOUT**

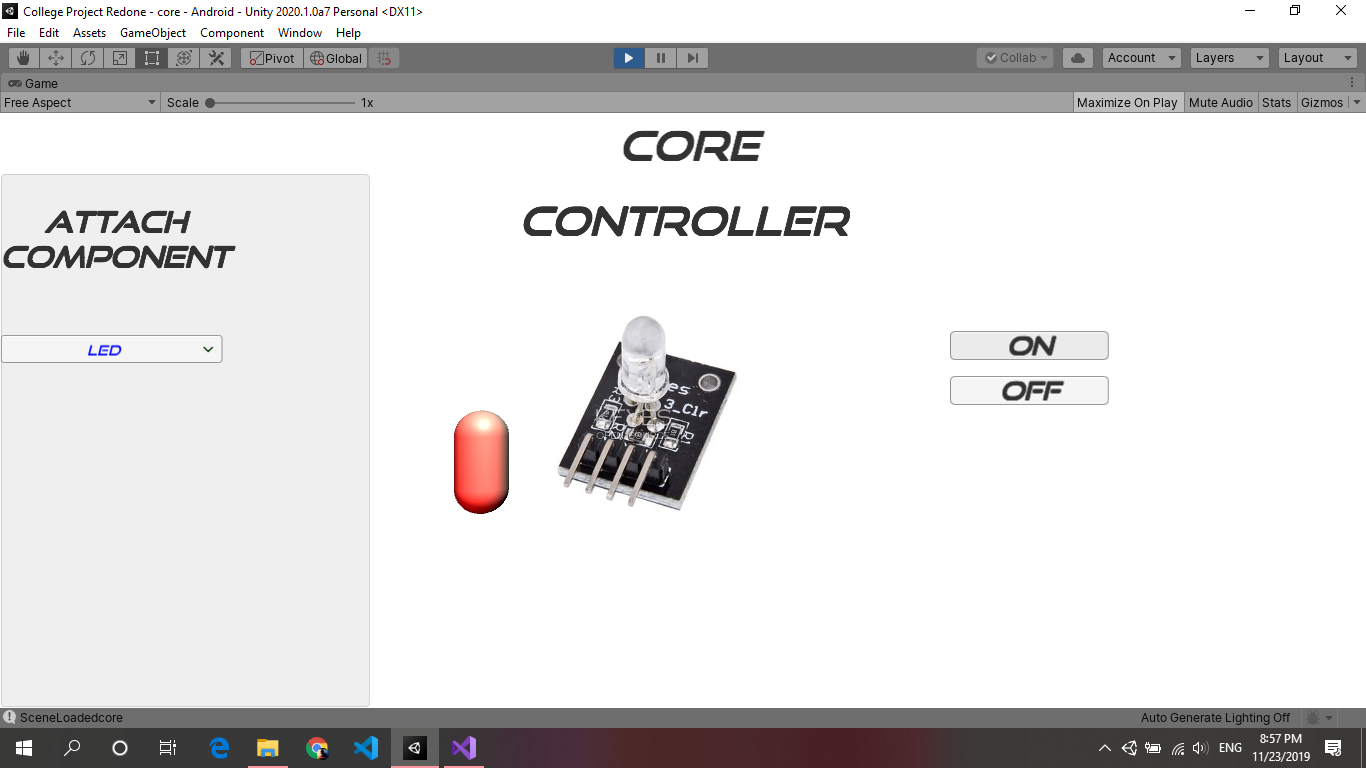
**5. Core Controller**

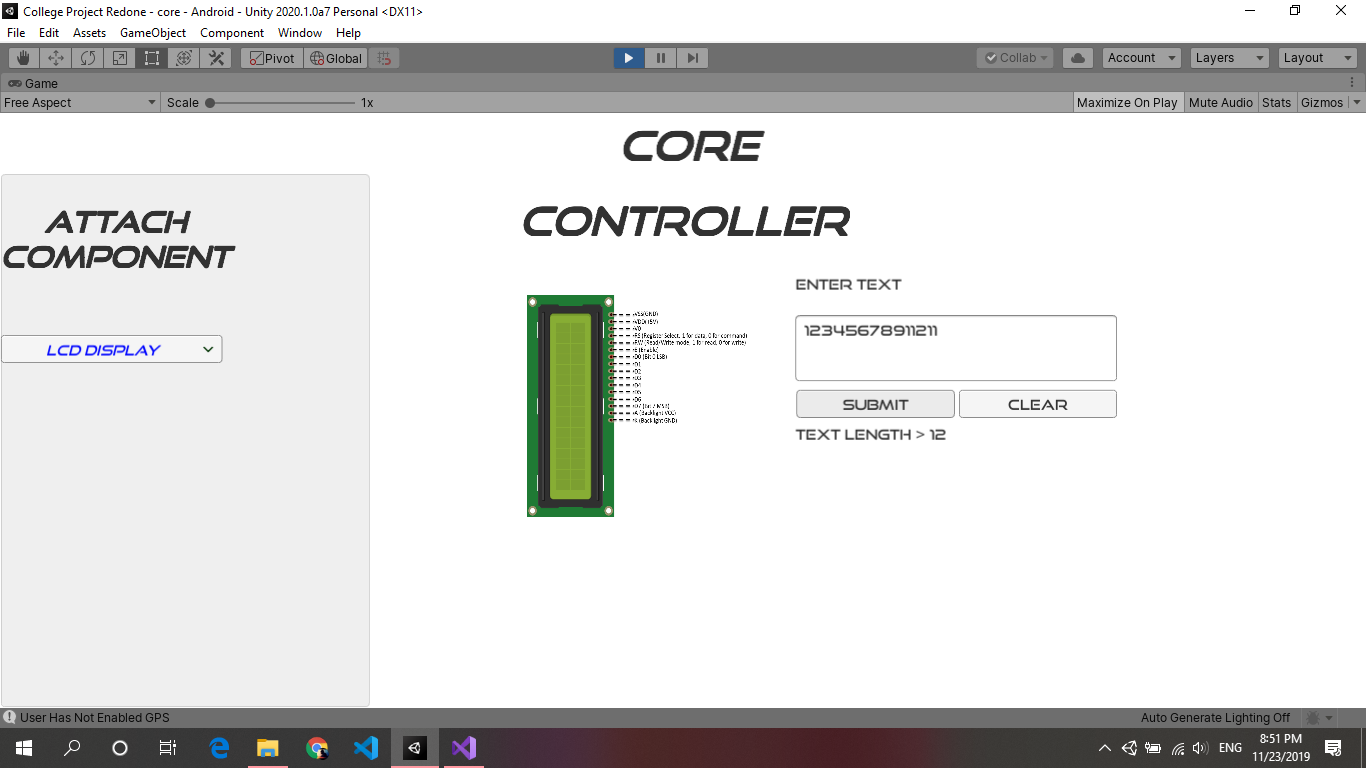
**5.1**

****

**5.2**

**5.3**

****

**5.4**

**5.5**

**G. TESTING**

**G.1 UNIT TESTING**

Testing for each unit in the software

The first module contains the component introduction where we tested if it displays the same information and same component image of the ComboBox value selected by the user.

In the Core module there are total 5 components

1. Buzzer – Test case for buzzer sound. We enabled it on start and provided buttons for stop and continue. Performs on button clicks respectively.
2. Camera – Testing if the system contains a camera sensor, if the sensor is found then we enable the sensor.
3. LCD Display – This IOT Component can only display 12 characters on the screen. We validated this test case if the user enters more than 12 characters it displays a message.
4. LED – Test case for On and OFF of LED light that is if the user clicks on ON button the LED light should get on and if clicks on OFF button the light should get off.
5. GPS – First we have checked the platform whether it is a system or android. Then next we tested if it is showing the correct Longitude and latitude of the current area and it updates as we move from one place to another.

**G.2 SYSTEM TESTING**

**For System Level:** All the Components Work with all given sensors. The GPS Sensor doesn’t work on system due to the Location can’t be manipulated on movement.

**For Android:**

The GPS Sensor works on android with other Scenes and core components.

**H. FUTURE ENHANCEMENT**

* More components can be added.
* More flexible UI can be created in AR for user engagement.
* Full-fledged android app can be created for IOT simulation for companion with system software.

1. **CONCLUSION**

The IOT simulation software provides the basic software for IOT implementation. This is helpful for learners to understand the simulation of IOT. It reduces the component cost required and it is faster on any provided system and also works on android devices.

The scope for the simulation purpose is limited to the fewer sensors in future enhancement AR application can be provided for easier simulation.

**J. BIBLIOGRAPHY**

* Learning C# by Developing Games with Unity 2019: Code in C# and build 3D games with Unity, 4th Edition
* Complete Unity 2018 Game Development: Explore techniques to build 2D/3D applications using real-world examples
* Unity.com/learn