**GROUP 3 : VRInGame**

**Interactive Virtual Reality**

**Submitted to : Dr Kewei Sha**

**Submitted On : May 4, 2018**

**Mentors : Dr Sheng Li**

**Dr. Zhandong Liu**

**Team Members : Siddharth Reddy Lucky**

**Vipasha Rana**

**Roshini Gnanamoorthy**

**Krishna Teja Talasila**

**Index**

1. **Abstract**
2. **Introduction**

**2.a. What is the problem?**

**2.b. What is the importance of the problem?**

**2.c. What is the solution?**

**2.d. What is the significance of the solution ?**

**3. Architecture Design**

**3.a. Sensor Module**

**3.b. DAQ Device  
 3.c. Signal Generation and Visualization**

**3.d. Virtual Reality Module**

**Abstract**

Imagine leaving a dull, stressful day behind and slipping into a magical world underwater. That’s the vision of BrEStim, a new intervention in the world of virtual reality where subjects are taken into a virtual environment leaving them calm and de-stressed. The Oculus headsets are used to immerse the subject in a peaceful virtual world that takes their focus off discomfort associated with medical problems and treatments.

The experience begins with collecting the subject’s breathing signal’s via a sensor. This short breathing exercise takes you to a world of water animals where a dolphin, chasing a puffer fish is controlled by the inspiratory and expiratory phases of respiration. The breathing signal will interact with the scene in VR simultaneously delivering an electrical stimulus when the inspiration reaches a preset target or threshold.

**Introduction**

**Problem Introduced :**

The previous implementation included the use of heavy designs and components. It can be difficult for the subject sometimes to make it to the hospital for the constant therapies. We needed a system which is more user friendly and modular.

**Why is the problem important ?**

This system implementation majorly focus on de-stressing and calming the patient’s while they are on a therapy to relieve the pain. The use of BreStim, ensures that the patient shows good results and significant improvement, as their tendency to take the electric stimulus increases while they are immersed in a virtual reality scene.

**Solution to the problem :**

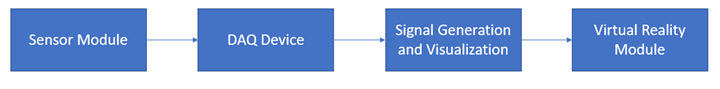
We have integrated the two platforms to satisfy the demands of our problem. The matlab platform calculates values that are above the preset target ie the threshold which triggers the dolphin in the scene to catch the puffer fish. The virtual reality scene is developed using Unity engine, which provides innovative tools and packages to make the scene more appealing to the subjects. MATLAB also communicates with the DAQ device to collect the breathing data of the patient as well as signalling the DAQ to deliver an electrical stimulus. This communication between MATLAB and DAQ is done using analog signals. At last both the matlab and the unity platform communicate using the socket communication implemented in the MATLAB.

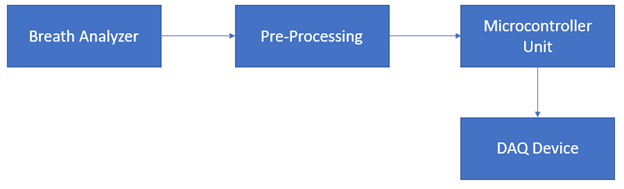
**Significance of the solution:**

Our system is made in order to satisfy these requirements:

1. Modularity
2. Scalability
3. Enables the swapping of the components easily when needed.

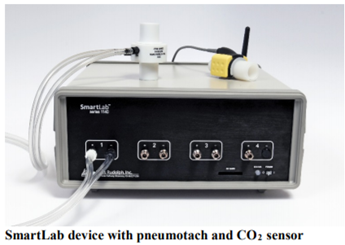
**ARCHITECTURE DESIGN:**

****

**SENSOR MODULE  
**

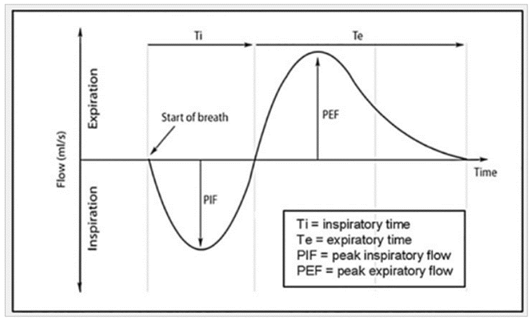
Respiration Sensor: This sensor is used to measure the inspiration rate of the subject i.e. the depth of a subject’s breath which is used to control the objects in a VR game is measured using this sensor.

**DAQ DEVICE:**

****

DAQ device with insight software is a flexible measurement system for use in making measurements of respiratory signals. It is a modular system that can be configured for a variety of voltage measurements. The base module consists of a main system circuit board that can accept up to four sensor modules or analog input modules as well as analog output modules.These output modules make it possible to output a voltage.. If a sensor is used to measure flow with a pneumotach it could be calibrated to work with several pneumotach each with a unique calibration. The software allows the user to select the calibration table to be used for the current setup. Once a certain breathing threshold is reached this DAQ device produces an output voltage which server as an electrical stimulus to the patient.

**Signal Generation and Visualization:**



The above figure represents the Inspiration Rate Waveform. In this graph, time is represented on the X-Axis and the Flow on Y-Axis. Flow is represented in ml/sec representing the volume of air taken in during the Inspiration phase and the volume of air sent out during the expiration phase. The gap between the start of one crest to the next represents one breath cycle. The above figure also shows various variables that are used to extract the data from the signal.

MATLAB collects the analog input signals from one of the channels and analyzes the data to form a waveform as shown in the figure above. When a crest of the waveform reaches a specific threshold MATLAB sends a signal to the DAQ to output a specific voltage on a specified analog output channel for a certain amount of time.

**MATLAB:**

clc

clear all

close all

%receive breathing signal from DAQ

s = daq.createSession('ni');  
 addAnalogInputChannel(s,'cDAQ1Mod4', 0, 'Voltage');

s.Rate = 8000

s.DurationInSeconds = 2;  
 s[data,time] = s.startForeground;  
 csvwrite(signal.dat,data)

plot(time,data);

xlabel('Time (secs)');  
 ylabel('Voltage')

th=0.28;

[a b] = uigetfile();

filename=[b a];

M = csvread(filename);

%M(:,1)= M(:,1)-th;

[x y]= size(M);

thline=th\*ones(x,1);  
%Creates communication between Unity and Matlab

t = tcpip('localhost',3002,'NetworkRole','Server');

fopen(t);

for i=101:x-101

plot(i,thline(i),'g.')

hold on  
%When the data of breath rate is above threshold graph marks with yellow point.

if M(i+1,1)<=th && M(i-1,1)>=th &&(i-yp)>=thd && flag == 1 && M(i,1)>th

'Threshold crossing detected'

point = [px py]

plot(px,py,'y\*')  
%Sends data packet “a” as an acknowledgement to unity

a='1catch1fish';

fwrite(t,a);

hold on

get (t, 'ValuesReceived')

%Signals DAQ to output electrical stimulus

devices = daq.getDevices  
 s = daq.createSession('ni');  
 addAnalogOutputChannel(s,'DeviceID',1,'current');  
 s.Rate = 100;  
 data = repmat (3, 1000);  
 data = data(:, 1);  
 plot(data, 'r')  
 hold on  
 grid on  
 xlabel('Datapoints')  
 ylabel('Amps')  
 legend('Data')  
 queueOutputData(s,data)  
 s.startForeground();

%Receives acknowledgement from unity

A = fread(t);

string = char(A);

disp(string);

elseif M(i,1)> th

plot(i,M(i,1),'r.')

else

plot(i,M(i,1),'b.')

end

hold on

pause(1\*10^-100)

end

Here is a list of tool boxes that need to be installed in MATLAB to ensure proper working of the DAQ device.

1. NI-DAQmx Support from Data Acquisition Toolbox
2. Instrument Control Toolbox
3. Signal Processing Toolbox
4. Data Acquisition Toolbox

**OCULUS RIFT - VIRTUAL REALITY MODULE**

There are basically two ways to interact with the reality:

Augmented reality that superimposes a computer-generated image on a user's view of the real world, thus providing a composite view. The components that can be used to achieve this are Intel smart glasses, hollow lens etc.

Virtual reality allows the user to enter a case scenario where it can interact with the real environment. The different devices that can be used are oculus rift, htc vive etc.

Several platforms are used now-a-days for game development which includes Unreal engine, Unity, Cryengine etc. Unreal engine is the most popular among the users as it is used mostly for graphical intensive games because of its advanced shaders and textures. Unity came into view when it made several updates and it is easy to handle and work with. We are making the use of unity, it acts as a controller. The implementation can be described using this flow chart: -

**UNITY:**

Unity is now regarded as one of the best game engines for developing VR, Desktop and Mobile applications. Due to its advanced pipelining, scheduling the developers are showing interest in working on this platform. We utilized the terrain engine provided by unity to generate a terrain from a mesh developed in C#. The code is as follows:

using UnityEngine;  
using UnityEditor;  
   
public class Object2Terrain : EditorWindow {  
   
 [MenuItem("Terrain/Object to Terrain", false, 2000)] static void OpenWindow () {  
   
 EditorWindow.GetWindow<Object2Terrain>(true);  
 }  
   
 private int resolution = 512;  
 private Vector3 addTerrain;  
 int bottomTopRadioSelected = 0;  
 static string[] bottomTopRadio = new string[] { "Bottom Up", "Top Down"};  
 private float shiftHeight = 0f;  
   
 void OnGUI () {  
   
 resolution = EditorGUILayout.IntField("Resolution", resolution);  
 addTerrain = EditorGUILayout.Vector3Field("Add terrain", addTerrain);  
 shiftHeight = EditorGUILayout.Slider("Shift height", shiftHeight, -1f, 1f);  
 bottomTopRadioSelected = GUILayout.SelectionGrid(bottomTopRadioSelected, bottomTopRadio, bottomTopRadio.Length, EditorStyles.radioButton);  
   
 if(GUILayout.Button("Create Terrain")){  
   
 if(Selection.activeGameObject == null){  
   
 EditorUtility.DisplayDialog("No object selected", "Please select an object.", "Ok");  
 return;  
 }  
   
 else{  
   
 CreateTerrain();  
 }  
 }  
 }  
   
 delegate void CleanUp();  
   
 void CreateTerrain(){   
   
 //fire up the progress bar  
 ShowProgressBar(1, 100);  
   
 TerrainData terrain = new TerrainData();  
 terrain.heightmapResolution = resolution;  
 GameObject terrainObject = Terrain.CreateTerrainGameObject(terrain);  
   
 Undo.RegisterCreatedObjectUndo(terrainObject, "Object to Terrain");  
   
 MeshCollider collider = Selection.activeGameObject.GetComponent<MeshCollider>();  
 CleanUp cleanUp = null;  
   
 //Add a collider to our source object if it does not exist.  
 //Otherwise raycasting doesn't work.  
 if(!collider){  
   
 collider = Selection.activeGameObject.AddComponent<MeshCollider>();  
 cleanUp = () => DestroyImmediate(collider);  
 }  
   
 Bounds bounds = collider.bounds;   
 float sizeFactor = collider.bounds.size.y / (collider.bounds.size.y + addTerrain.y);  
 terrain.size = collider.bounds.size + addTerrain;  
 bounds.size = new Vector3(terrain.size.x, collider.bounds.size.y, terrain.size.z);  
   
 // Do raycasting samples over the object to see what terrain heights should be  
 float[,] heights = new float[terrain.heightmapWidth, terrain.heightmapHeight];   
 Ray ray = new Ray(new Vector3(bounds.min.x, bounds.max.y + bounds.size.y, bounds.min.z), -Vector3.up);  
 RaycastHit hit = new RaycastHit();  
 float meshHeightInverse = 1 / bounds.size.y;  
 Vector3 rayOrigin = ray.origin;  
   
 int maxHeight = heights.GetLength(0);  
 int maxLength = heights.GetLength(1);  
   
 Vector2 stepXZ = new Vector2(bounds.size.x / maxLength, bounds.size.z / maxHeight);  
   
 for(int zCount = 0; zCount < maxHeight; zCount++){  
   
 ShowProgressBar(zCount, maxHeight);  
   
 for(int xCount = 0; xCount < maxLength; xCount++){  
   
 float height = 0.0f;  
   
 if(collider.Raycast(ray, out hit, bounds.size.y \* 3)){  
   
 height = (hit.point.y - bounds.min.y) \* meshHeightInverse;  
 height += shiftHeight;  
   
 //bottom up  
 if(bottomTopRadioSelected == 0){  
   
 height \*= sizeFactor;  
 }  
   
 //clamp  
 if(height < 0){  
   
 height = 0;  
 }  
 }  
   
 heights[zCount, xCount] = height;  
 rayOrigin.x += stepXZ[0];  
 ray.origin = rayOrigin;  
 }  
   
 rayOrigin.z += stepXZ[1];  
 rayOrigin.x = bounds.min.x;  
 ray.origin = rayOrigin;  
 }  
   
 terrain.SetHeights(0, 0, heights);  
   
 EditorUtility.ClearProgressBar();  
   
 if(cleanUp != null){  
   
 cleanUp();   
 }  
 }  
   
 void ShowProgressBar(float progress, float maxProgress){  
   
 float p = progress / maxProgress;  
 EditorUtility.DisplayProgressBar("Creating Terrain...", Mathf.RoundToInt(p \* 100f)+ " %", p);  
 }  
}

The asset packs that are used are to be organized into assets for easier access:

Here is a list of all the assets that were utilized in the project.

1. Prefab Brush: <https://assetstore.unity.com/packages/tools/terrain/prefab-brush-21321>
2. Corals Pack: <https://assetstore.unity.com/packages/3d/environments/coral-pack-14389>
3. Dolphin: <https://assetstore.unity.com/packages/3d/characters/animals/dolphins-26937>
4. Crucian Carp Fish: <https://assetstore.unity.com/packages/3d/characters/animals/free-cartoon-crucian-carp-46132>
5. Nature starter Kit 2: Preloaded by unity on startup.

A prefab brush can be used to paint the required terrain texture on to the generated terrain. Lighting values can be changed to Real time global illumination for faster loading, job scheduling. Once the direction lighting and the environment are adjusted to linking, water assets is unity is used to generate water. A projector can be placed on the water surface to generate caustics. The models can be placed in the environment accordingly. The models include the coral sea bed, dolphin, the fish, ornaments for decoration such as Sunken ship and a closed treasure chest to make the environment more interesting. The script to generate caustics is as follows:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Animated\_Projector : MonoBehaviour

{

public float fps = 30.0f;

public Texture2D[] frames;

private int frameIndex;

private Projector projector;

// Use this for initialization

void Start()

{

projector = GetComponent<Projector>();

NextFrame();

InvokeRepeating("NextFrame", 1 / fps, 1 / fps);

}

//Refers to the next frame to be projected

void NextFrame()

{

projector.material.SetTexture("\_ShadowTex", frames[frameIndex]);

frameIndex = (frameIndex + 1) % frames.Length;

}

}

**Fishes, spawn, movement management**

To make the interaction look more realistic, we had to implement the concepts of Artificial Intelligence. Unity provides an AI manager to manage all the objects, group them and manage their characteristic to achieve true random movement. The FishSpawner.cs script is attached to the lake (All the objects are spawned in this area).

The objects that have the tag “Fishes” will be generated at a random range at a random point, with a random time difference between spawns. When each fish is spawned the AI manager attaches another script that instructs the fishes to move along all the waypoints in the lake. The script that enables movement is as follows:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class FishAIMove : MonoBehaviour {

// Declare variables for AI Spawner manager script

private Fish\_AISpawner m\_fishAIManager;

// Declare variables for moving and turning

private bool m\_fishhasTarget = false;

private bool m\_fishIsTurning;

// Variable for current waypoint

private Vector3 m\_fishWayPoint;

private Vector3 m\_fishlastWayPoint = new Vector3(0f, 0f, 0f); // This is to check if the new waypoint is same as the last one

// These variables will be used to set the animation speed

private Animator m\_fishAnimator;

private float m\_fishSpeed;

// Adding and checking with a collider to make the fish behaviour look more realistic

private Collider m\_fishCollider;

// Use this for initialization

void Start ()

{

m\_fishAIManager = transform.parent.GetComponentInParent<Fish\_AISpawner>();

m\_fishAnimator = GetComponent<Animator>();

Fish\_SetupNPC();

}

void Fish\_SetupNPC()

{

//Randomly sacale each NPC

float m\_fishScale = Random.Range(5f, 9f);

transform.localScale += new Vector3(m\_fishScale \* 7f, m\_fishScale, m\_fishScale);

// Checking to see if the prefab has a collider

// Add it even if the child has a collider.

if(transform.GetComponent<Collider>() != null && transform.GetComponent<Collider>().enabled == true)

{

m\_fishCollider = transform.GetComponent<Collider>();

}

else if(transform.GetComponentInChildren<Collider>() != null && transform.GetComponentInChildren<Collider>().enabled == true)

{

m\_fishCollider = transform.GetComponentInChildren<Collider>();

}

}

// Update is called once per frame

void Update ()

{

if(!m\_fishhasTarget)

{

m\_fishhasTarget = Fish\_CanFindTarget();

}

else

{

// Rotate the NPC to face the waypoint - fish

Fish\_RotateNPC(m\_fishWayPoint, m\_fishSpeed);

// Move the NPC in a straight line toawards the target waypoint

transform.position = Vector3.MoveTowards(transform.position, m\_fishWayPoint, m\_fishSpeed \* Time.deltaTime);

//Check if the fish collided, if yes, look for a new waypoint

//FishCollidedNPC();

}

// If NPC reaches the waypoints reset the targets

if(transform.position == m\_fishWayPoint)

{

m\_fishhasTarget = false;

}

}

bool Fish\_CanFindTarget(float fish\_start = 15f, float fish\_end = 25f)

{

m\_fishWayPoint = m\_fishAIManager.RandomWaypoint();

if(m\_fishlastWayPoint == m\_fishWayPoint)

{

// Get a new waypoint

m\_fishWayPoint = m\_fishAIManager.RandomWaypoint();

return false;

}

else

{

// Function accepted setting the new waypoint as the last waypoint

m\_fishlastWayPoint = m\_fishWayPoint;

// Get random speed for movement and animation

m\_fishSpeed = Random.Range(fish\_start, fish\_end);

m\_fishAnimator.speed = m\_fishSpeed;

// Set boolen value to true as we found a waypoint

return true;

}

}

// Function to rotate the fish to a valid direction

void Fish\_RotateNPC(Vector3 fish\_waypoint, float fish\_currentSpeed)

{

//get random speed up for the turn

float fish\_TurnSpeed = fish\_currentSpeed \* Random.Range(1f, 3f);

//get a new direction to look at the target

Vector3 fish\_LookAt = fish\_waypoint - this.transform.position;

transform.rotation = Quaternion.Slerp(transform.rotation, Quaternion.LookRotation(fish\_LookAt), fish\_TurnSpeed \* Time.deltaTime);

}

// method to check if the fish collided with another fish or a body

void FishCollidedNPC()

{

RaycastHit fish\_hit;

if(Physics.Raycast(transform.position, transform.forward, out fish\_hit, transform.localScale.z))

{

if(fish\_hit.collider == m\_fishCollider | fish\_hit.collider.tag == "fish\_waypoint")

{

return;

}

// Randomizing the chance where the fish will change direction after bumping

int randomFish\_num = Random.Range(1, 100);

if(randomFish\_num < 40)

{

m\_fishhasTarget = false;

}

// Loggin the various colliders hits.

Debug.Log(fish\_hit.collider.transform.parent.name + " "+ fish\_hit.collider.transform.parent.position);

}

}

// The Fish initially spawn at a fixed location,

// Using this function will make it truly random.

Vector3 Fish\_getWayPoint(bool fish\_isRandom)

{

// if fish\_isRandom true then get a random position location

if(fish\_isRandom)

{

return m\_fishAIManager.RandomPosition();

}

// This will be used to get a random waypoint

else

{

return m\_fishAIManager.RandomWaypoint();

}

}

}

**Dolphin - Spawn, Movement management**

The model of dolphin is placed in the environment with the Oculus rift camera rig following it. Later this is the point where the user will be viewing from. The OVR camera is made to follow the dolphin. A script is attached to the dolphin that enables movement. The script is as follows:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System.Linq;

public class DolphinAI\_SingleObject

{

public string DolphinGroupName { get { return m\_DolphinAIGroupname; } }

//public GameObject Dolphin\_objectPrefab { get { return m\_Dolphinprefab; } }

[Header("Dolphin AI Stats")]

[SerializeField]

private string m\_DolphinAIGroupname;

// Create array from the new class

// Uncomment the below for dolphin group customization

// Constructor for accessing AI values above

public DolphinAI\_SingleObject(string Dolphin\_Name)//, GameObject fish\_Prefab)

{

this.m\_DolphinAIGroupname = Dolphin\_Name;

//this.m\_Dolphinprefab = fish\_Prefab;

}

}

// You can use this script if you want to generate multiple dolphins.

public class DolphinAI\_Spawn : MonoBehaviour {

// List that holds all the waypoints for the dolphin to move

public List<Transform> Dolphin\_Waypoints = new List<Transform>();

// Use this for initialization

void Start ()

{

DolphinGetwayPoints();

InvokeRepeating("DolphinSpawnNPC", 3.0f, 10f);

}

// Update is called once per frame

void Update () {

}

void DolphinGetwayPoints()

{

// List to look through all the nested children.

Transform[] dolphin\_waypointList = transform.GetComponentsInChildren<Transform>();

for (int i = 0; i < dolphin\_waypointList.Length; i++)

{

// Initially for checking we are using the same waypoints as the fishes

if (dolphin\_waypointList[i].tag == "fish\_waypoint")

{

// Adding values of waypoints to the list

Dolphin\_Waypoints.Add(dolphin\_waypointList[i]);

}

}

}

void DolphinSpawnNPC()

{

// Need a better way to fing the GameObject for the AI

GameObject DolphinVal = GameObject.Find("Dolphin\_Master");

Quaternion randomRoatation = Quaternion.Euler(Random.Range(-20, 20), Random.Range(0, 360), 0);

DolphinVal.AddComponent<DolphinAI\_Movement>();

}

public Vector3 RandomWaypoint()

{

int Dolphin\_randomWP = Random.Range(0, (Dolphin\_Waypoints.Count - 1));

Vector3 Dolphin\_randomWaypoint = Dolphin\_Waypoints[Dolphin\_randomWP].transform.position;

return Dolphin\_randomWaypoint;

}

}

All these scripts are executed parallelly upon starting of the project, a socket connection is established to MATLAB. Once a trigger is fired an acknowledgement is sent to unity. A subprocess ie, a coroutine is triggered instructing the dolphin to go and catch the fish. Once the fish is caught an acknowledgement is sent back to MATLAB and a counter value is incremented to keep track of the patients score.

**Milestones and Contributions of team members:**

1. Being new to the platform we had to get ourselves up to speed on Unity. The project demanded implementing an underwater scenario which had to look appealing to the user. So we had to design the scene from ground up.
2. To make the movement realistic we had to implement Nav mesh and Rigid body movements using Artificial Intelligence. Being beginners in this domain we had to conduct extensive research on this.
3. We had to study the basic animator class so as to define the swimming movements of the dolphin and fishes.
4. We had to establish socket communication to send packets of acknowledgement data from MATLAB to Unity.
5. We also had to ensure that there was no delay in packet transmission as the delay would affect the animation of the dolphin.
6. We had to make sure the animation was quick so that the next acknowledgement wouldn't have to wait to show an affect.
7. We had to study the NI instruments tool boxes in order to create the input/output connections with the DAQ.
8. We had to create session interactions between matlab and DAQ.
9. We had to analyze the channel configurations of DAQ to correspond it to the channels on which MATLAB takes input or produces the output.
10. We had to make the code as customizable as possible so that it can behave as a new scene on demand.
11. Meeting was one of the milestones where in we had to arrange a meeting in such a way that everyone could accommodate the same in their schedule.

**Conclusion and future work:**

1. The system is designed with scalability and modularity in mind.
2. The code is highly customizable and well documented.
3. The models can be upgraded to more realistic high texture models to make the scene more realistic.
4. The caustics generator can be upgraded to surround the dolphin to make the caustics more evident.
5. Camera movement can be smoothed so that the patients do not tend to get sea sick while in Virtual Reality.
6. This project was taken up to keep the patients engaged in the game, relieve their stress.
7. There is always a scope of improving the project and the AI algorithm to make the movements and grouping of the fish AI.
8. Include animations to the fishes to enable realistic swimming and animation of dolphin eating the fish.
9. Higher definition models can be swapped for the same to make the scene more realistic.
10. The breathing signal sensor can be made wireless to prevent hassle to the patient going everytime to the hospital to get the treatment.

**References:**

[1] Unity3d.com, 'Unity Manual: for water, Transform, Rigidbody movement', 2018. [Online]. Available: https://unity3d.com/learn/tutorials. [Accessed: 03-Feb-2018].  
[2] Rudolphkc.com, 'To know about breath sensor and DAC', 2018. [Online]. Available: http://www.rudolphkc.com/. [Accessed: 20-Mar-2018].  
[3] Oculusstudios.com, 'To use oculus utilities with Unity', 2018. [Online]. Available: http://oculusstudios.com/work/. [Accessed: 25-Mar-2018].  
[4] S. Attaway, Matlab: A Practical Introduction to Programming and Problem Solving. (2nd;2; ed.) 2012;2014;.  
[5] W. Goldstone, Unity Game Development Essentials. 2009.  
[6] Mathworks.com, 'To learn about Matlab', 2018. [Online]. Available: https://www.mathworks.com/academia.html?s\_tid=gn\_acad#learn-basics. [Accessed: 25-Feb-2018].