**CHAPTER I**

**INTRODUCTION**

The fire fighter faces hazards and dangers during the course of their rescue. Some physical hazards are hard to eliminate like injuries from back drafts or loud noises due to burn. But we can support them technically and that’s the objective of this project. The PYRO HEAD-GEAR is a design with hardware and software support that could effectively assist firefighters. It allows the fire fighters to know about their physical conditions in their environment.

The PYRO HEAD-GEAR works on cross platform support. Multiple platforms

are used here and the platform can be switched based on their requirement and usability.

Fire fighters face serious risks on the job. They face high temperatures and high levels of carbon monoxide (CO) and other toxic risks in the areas around fires. It is important to know about the temperature level and carbon monoxide level over the environment.

Due to the chemical make-up of many manufactured materials, today’s fires reach hotter temperatures faster, flashovers occur more rapidly and the resulting smoke is much more toxic.

As a result of these factors, firefighters have been dying at a greater rate than ever before from inhaling toxic chemical Understanding CO is especially crucial to today’s fire service, because the smoke that firefighters were exposed to 20 or 30 years ago is not the same as it is today. Exposure to higher levels of CO, when combined with HCN, will mimic the signs and symptoms of cardiac-related emergencies. Acute exposure to HCN can result in symptoms such as weakness, headache, confusion, vertigo, dyspnea and, occasionally, nausea and vomiting. Respiratory rate and depth usually increase at the onset, and will eventually cause the victim to gasp for breath. Coma and convulsions occur in some cases. If a firefighter gets to the point where they lose color and become ashen, or cyanosis is present, it usually means that respirations have ceased or have been inadequate for an extended amount of time.

**1.1 HAZARDOUS GASES**

The Hazardous Gases has certain levels that should not be exceeded in the air content, if that exceeds it becomes very dangerous. By knowing the level of these gases appropriate action could be taken by the fire fighter during the rescue mission. Out of this many dangerous gases, the most dangerous and more frequently occurring gas during fire accident is Carbon Monoxide.

* + 1. **CARBON MONOXIDE**

In this project, Detecting Carbon Monoxide is given importance because it is considered to be the most dangerous during fire accident.

Carbon Monoxide is a colorless, odorless, tasteless and poisonous gas that is slightly less dense than air. It is toxic when encountered in concentrations above about 35 ppm and people should avoid breathing such air as it causes may lead dangerous health issues. It is produced by the incomplete burning of solid, liquid, and gaseous fuels. Appliances fueled with natural gas, liquefied petroleum (LP gas), oil, kerosene, coal, or wood may produce CO. Burning charcoal produces CO.

Depending on the level of CO and length of exposure, you may experience any one or more of the following symptoms:

• Headache

• Dizziness

• Weakness

• Clumsiness

• Nausea and vomiting

• Quick irregular heartbeat

• Chest pain

• hearing loss

• Blurry vision

• Disorientation

• Confusion seizures

Certain high level of Carbon Monoxide may cause more effects to humans or to any hemoglobic animal. When it exceeds 1600 ppm, Death occurs when the person is exposed to it for one or two hours. More dangerous it becomes when the level exceeds 6000 ppm, Death occurs in few minutes of exposure.

* + 1. **OTHER GASES**

The other dangerous gases include Hydrogen Cyanide, Sulphur dioxide, phosgene, acrolein etc. Fire fighters are also prone to exposure of these gases which are injurious to health.

Hydrogen Cyanidesometimes called prussic acid is an organic compound. It is a colorless, extremely poisonous liquid that boils slightly above room temperature, at 25.6 °C. A hydrogen cyanide concentration of 300 mg/m3 in air will kill a human within 10–60 minutes. A hydrogen cyanide concentration of 3500 [ppm](https://en.wikipedia.org/wiki/Parts_per_million) (about 3200 mg/m3) will kill a human in about 1 minute. The toxicity is caused by the cyanide ion, which halts [cellular respiration](https://en.wikipedia.org/wiki/Cellular_respiration) by acting as a [non-competitive inhibitor](https://en.wikipedia.org/wiki/Non-competitive_inhibitor) for an enzyme in mitochondria called [cytochrome c oxidase](https://en.wikipedia.org/wiki/Cytochrome_c_oxidase).

Depending on the level of HCN and length of exposure, you may experience any one or more of the following symptoms:

* Progressive hypoxia including headache, anxiety, agitation, confusion, lethargy, seizures and coma.
* Cardiovascular effects - Initially bradycardia and hypertension may occur followed by hypotension and tachycardia. The terminal event is consistently bradycardia and hypotension.
* Respiratory - Initial patient findings may include increased respiratory rate, shortness of breath and chest tightness. With progression of poisoning, respirations become slow and gasping. Central cyanosis may or may not occur. Pulmonary edema may occur.
* GI toxicity following ingestion of cyanide may occur. This may include abdominal pain, nausea and vomiting.
* Skin - A cherry red skin color may be present as the result of increased venous hemoglobin oxygen saturation. Cyanide does not directly cause cyanosis. If present, it is secondary to shock.
* Ocular - Direct contact to liquid cyanide can result in eye irritation and swelling.
* Children and pregnant women are much more vulnerable than adults to cyanide agent toxicity.

Sulphur dioxide is a chemical compound which is toxic in nature with a pungent and irritating smell. It is also produced during fire accidents which may cause adverse health issues on breathing.

Thus in Pyro Head-Gear the hazardous gas is sensed using a sensor and it is displayed on a head up display which gives a fire fighter a clear idea about the situation. Thus the fire fighter can act accordingly.

* 1. **TEMPERATURE AND LOCATION**

When a fire fighter enters a building on a rescue mission, the temperature of the surrounding is required for him.

Thus the temperature is measured and it is also displayed in the head up display.

If a fire fighter needs an assistance of another fire fighter during a rescue or firefighting mission, the location of the fire fighter is required in order to assist him. It is been provided in the Pyro Head-Gear by means of google map which will be of a greater use. The location will be displayed in three dimension in the helmet using OLED. The Pyro Head-Gear has the potential to increase the fire fighter's safety and make their work more efficient.

**CHAPTER II**

**EXISTING SYSTEM**

When a fire fighter enters a scene of fire accident, he is exposed to many hazardous gases like carbon monoxide, benzene, hydrogen chloride, hydrogen cyanide, sulphur dioxide, dichlorofluoromethane etc., Now in the current scenario there is no real means for him to know the levels of the hazardous gases in the place of the accident and so if he wants to know the level of those gases, he has to use individual sensor module.

If a fire fighter wants to communicate with a co fire fighter in a rescuing operation, the equipment usually used is a walkie talkie or cell phones. But there is no form of aid is given to know the exact position of the co fire fighters. May be the other fire fighter can give directions or open the Google maps to know the location but given the circumstances it is difficult. Pyro Head-Gear provides solution to these problems.

However, few papers are available on displaying a map on a head mount display.Typical applications of these Head Mounted Displays in the area of fire fighting are to display thermal imaging data or tactical information such as maps or navigation information which is two dimensional in nature.

More generic version of head mounted display which is based on a data acquisition system mounted on helmet for displaying environmental data and images in critical environments, particularly areas at risk of accident or in which an accident has occurred that enable sensors data fusion, in a networked integrated solution. This proposed solution exploits a distributed multi-sensor architecture for providing valuable information to human operators such as fire fighters, security etc. This uses thermal camera for the purpose of displaying images.

**CHAPTER III**

**PROPOSED SYSTEM**

**3.1 INTRODUCTION**

Pyro Head-gear aims at providing assistance to the modern age fire-fighters by augmented reality technology. The Augmented reality allows the software to interact with the visual perspective of the fire-fighter and providing him updates of his environment. GPS allows him to keep track of his team-mates and vice-versa.

The Pyro Head-gear consists of 3 modules. A three Dimensional AR-map which allows the user to track his team, an embedded system which gets the sensor input and transfer it to the main processor, the HUD which integrates the data from the other two modules and renders an AR display.

**3.2 AUGMENTED REALITY MODULE**

The 3-Dimentional map is created with 3Ds MAX and AR Toolkit and uses GPS co-ordinates for its orientation. To send and receive the co-ordinates between the team, an android APK is utilized. The APK is designed on android platform. The GPS data is collected from the android device and uploaded to a SQL data base. The collected data is then downloaded in the main processor. A custom algorithm is then used to map the co-coordinates. The 3D map’s abstract elements are designed in 3DS max. The Radar planes along with field marker models are developed and they are exported into the Unity software.

**3.2.1 LOCATION SHARE APK**

In the android studio the user lists are created. Here we have three users’ s001, s002 and s003. The user gives the ID on the home page of the application to login. The user location is sent to an online database where we store all the users’ Co-ordinates. The network provider manages the work between the application and the online data base. To access our data in a more convenient way, SQL server is used.

**3.2.2 AR TOOLKIT APK**

The Android SDK module in Unity Engine provided access the camera which collects feed and processes it in the AR tool kit module. A special set of markers known as a fiducial marker or a Glyph is used to know the positioning with respective to the camera. With this, the models exported are rendered

**3.2.3 IMAGE PROCESSING**

Optical Glyph tracking extensively uses image processing to track the fiducial marker or a glyph. A fiducial marker is a special type of marker which could only be recognized by the image processing software. The Glyph is uploaded while developing the software.



**FIG 3.1:** Glyph Marker

At frame updates, the software compares the sample image with the reference. This way, whenever the software recognizes the glyph, it examines the positioning of the glyph. The positioning includes the angle, tilt and the size of it. The glyph acts as a reference for the software to interact with the real world.

**3.2.4 IMAGE RENDERING**

The data from the glyph acts as a reference for the software to interact with the real world and this way, the orientation and model developed or downloaded is rendered with reference to the glyph. The reference includes the size and the angle of the render.

The movements of the Team mates are recognized with the GPS positioning and they are used to orient with respective to the First-person.

Thus by co-ordination both the android applications, cross platform data transfer is possible. Cloud server instead of SQL servers can further increase the performance.

**3.3 EMBEDDED SYSTEM MODULE**

The Embedded network uses a Linux based RTOS patch for real time data reception and an ADB protocol for data transfer. The Raspberry Pi provides hardware support. It receives the data from MQ7 and DB18S20 sensors. The MQ7 is uses to know the percentage of CO2 in the surroundings and DB18S20 is a temperature sensor. The feed from these sensors are received from the digital input pins of the raspberry. The Raspbian Kernel is re-programmed by a Xenomai patch to work as a Real-time OS providing fast updates on the transmission and reception data. The Linux uses shell programming coded in Python and the sensor data is saved in a XML file. This XML file is transferred to the main Memory by the ADB and then utilized by Unity.

**3.4 FILE TRANSFER**

The file transfer is done by using Extensible markup language(XML) and Android Debug Bridge(ADB).

**3.4.1 EXTENSIBLE MARKUP LANGUAGE**

The usage of XML is very advantages since it had a good support with Unity game engine and also it uses standard web protocols and this allows the developers to include larger data during transfer with less update rates. XML also has the advantage of all the web support features and hence standard TCP can be replaced by advanced database file transfer techniques such as cloud. Cloud servers could be used for both file transfer and also maintaining a report log for every fire safety incident.

**3.4.2 ANDROID DEBUG BRIDGE**

File transfers are made in the most universal protocols. They are implemented in such a way that they are not platform specific. Although the Android Debugging Bridge plug-in are added to the Linux Kernel, it uses standard TCP for file transfer and any devices other than Android doesn't require any other plug-in to be interfaced with it. This is a major advantage in developing cross platform modules.

**3.5 DISPLAY MODULE**

Finally the HUD is designed in Unity game engine which uses C# programming language and this is projected in a display which acts as a see through for the fire fighters. Stereoscopic render is possible via Unity Engine.

**3.6 BLOCK DIAGRAM**

The system consists of two processors. The axillary processor provides hardware support. It is an embedded system operating on a Real-time RTOS. Sensor modules could be interfaced with an interrupt management system in the auxiliary processor. Raspberry pi is used since it has digital input pins and could be directly interfaced with the sensors. The main processor is a user side interface. It supports the concept of customizable UI.

**CAMERA**

**MAIN PROCESSOR**

**DISPLAY**

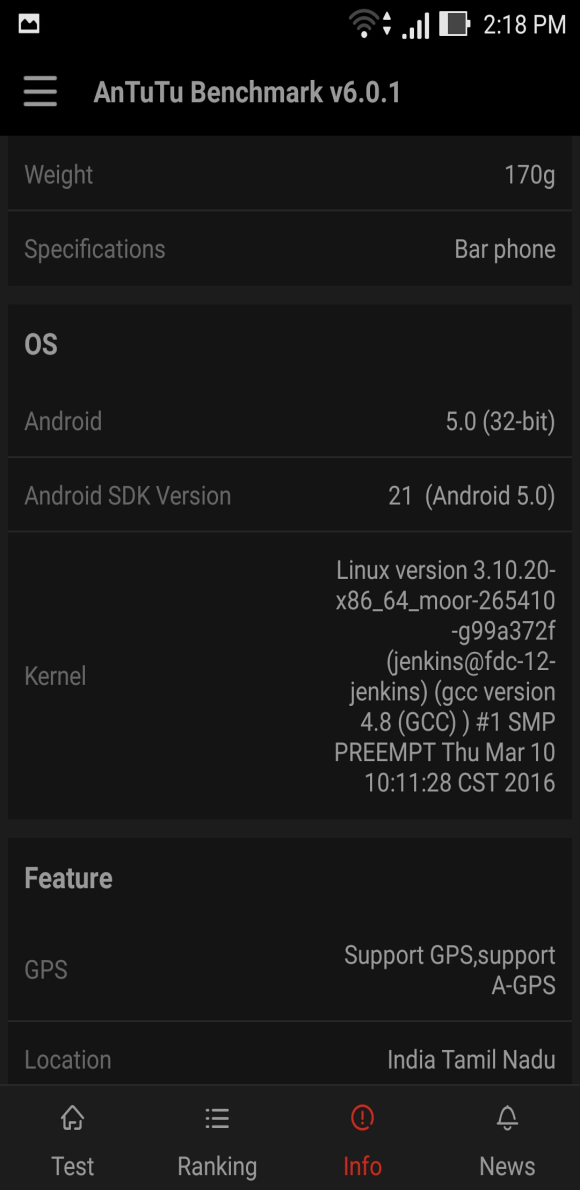
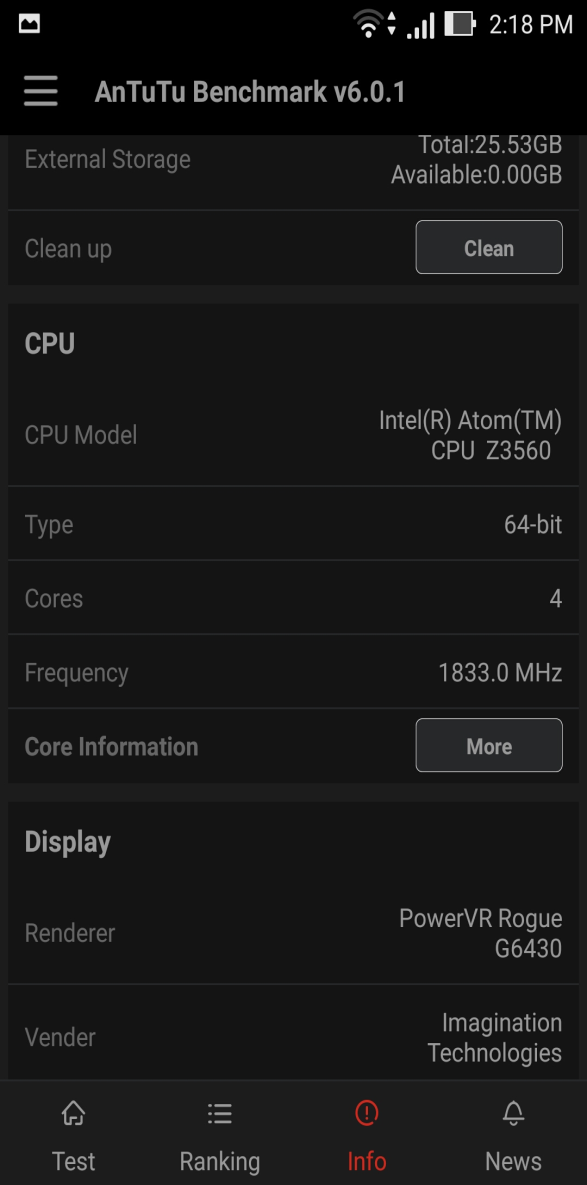
**GPS**

**AUXILIARY PROCESSOR**

**SENSORS**

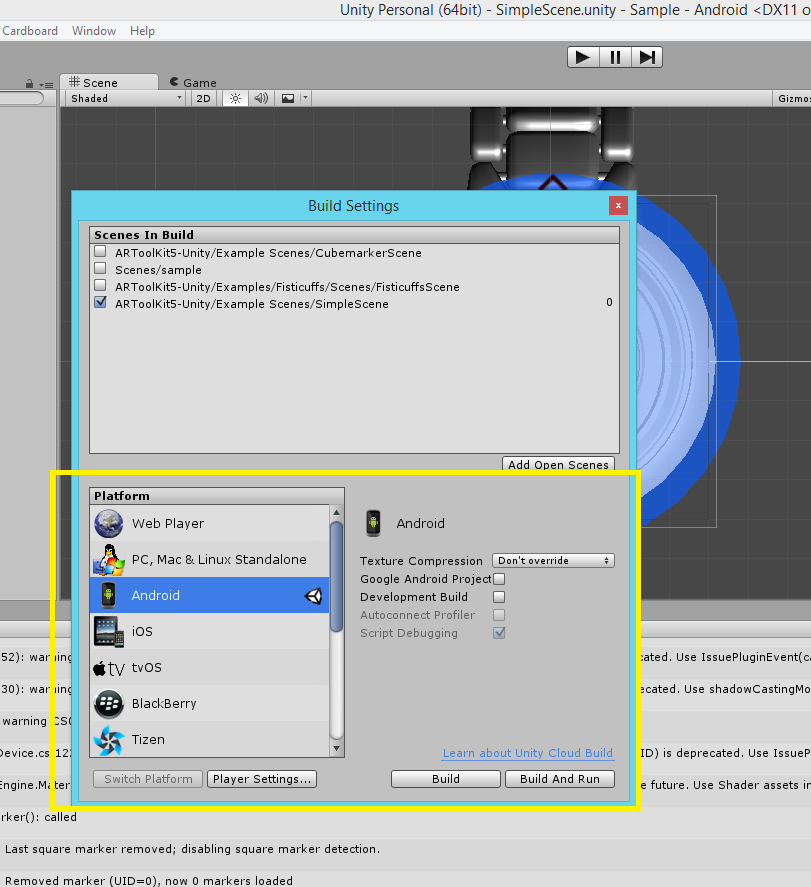
**FIG 3.2:** Block Diagram of Pyro Head-Gear

The UI design for HUD is also modular. The UI could be reprogrammed to the Fire Fighter’s convenience. This is run by the main processor(Intel® Atom™ CPU Z3560). It also renders the 3d models in AR. The gps data is transferred to the main processor and a virtual three dimensional map is rendered. The sensor data is transferred from the auxiliary processor to the main processor via TCP or USB protocol. Camera unit helps the main processor interact with the environment on real time. This is the primary source for display and could be replaced by thermal image processing. The display is a LED unit providing high definition real time camera feed and could also be replaced by a transparent OLED display. The final build and player settings are flexible enough to be built on any working platform which includes web player. Unity allows latest OS builds. This is one of the key features of the Pyro Head-Gear.

  
**FIG 3.3:** Main Processor and Operating System Specifications

The UI of the Pyro Head-Gear is also customizable via a standard free version of the software. Modules such as an inbuilt compass and other image processing techniques could be plugged in to support the fire fighter if necessary.

Although android is the most suited platform for the build because the device encourages developers by allowing a reprogrammable kernel with complete hardware support. All the sensors in the phone could be used and all of the processing power could be utilized by the developers.



**FIG 3.4:** Unity Build Settings

**CHAPTER IV**

**HARDWARE DESCRIPTION**

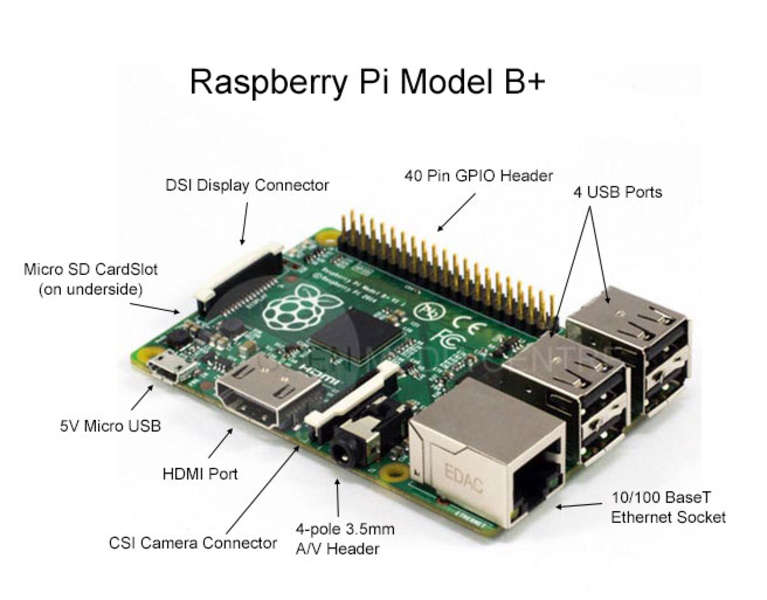
The Hardware part of Pyro Head-Gear consists of Raspberry PI and the sensors. Raspberry PI is used for interfacing the sensors with the main processor. The sensors are used here are MQ7 sensor which is used for sensing Carbon Monoxide and DB18S20 sensor which is a temperature sensor gives the temperature reading of the surroundings.

**4.1 RASPBERRY PI**

The Hardware part is managed by Raspberry Pi B2 with which the sensors are interfaced. Raspberry PI is a series of single board computers developed by Raspberry Pi Foundations. All raspberry models feature a Broadcom system on a chip (SOC) which include an ARM compatible CPU and an on chip Graphics processing unit GPU. CPU speed range from 700 MHz to 1.2 GHz for the Pi 3 and on board memory range from 256MB to 1GB RAM.

Secure Digital SD cards are used to store the operating system and program memory in either the SDHC or MicroSDHC sizes. Most boards have between 1 and 4 USB slots, HDMI and composite video output, and a 3.5mm phono jack for audio. Lower level output is provided by a number of GPIO pins which support common protocols like I2C. Some models have an RJ45 Ethernet port and the Pi 3 has on board Wi-Fi 802.11n and Bluetooth. The main reason for using Raspberry is its Real time OS compatibility.The Raspberry Pi hardware has evolved through several versions that feature variations in memory capacity and peripheral-device support.The [system on a chip](https://en.wikipedia.org/wiki/System_on_a_chip) (SoC) used in the first generation Raspberry Pi is somewhat equivalent to the chip used in older [smart phones](https://en.wikipedia.org/wiki/Smartphone) (such as[iPhone](https://en.wikipedia.org/wiki/IPhone), [3G](https://en.wikipedia.org/wiki/IPhone_3G), [3GS](https://en.wikipedia.org/wiki/IPhone_3GS)). The Raspberry Pi is based on the [Broadcom](https://en.wikipedia.org/wiki/Broadcom)BCM2835 SoC, which includes an 700 [MHz](https://en.wikipedia.org/wiki/Hertz) [ARM11](https://en.wikipedia.org/wiki/ARM11)76JZF-S processor, [Video Core](https://en.wikipedia.org/wiki/VideoCore) IV [graphics processing unit](https://en.wikipedia.org/wiki/Graphics_processing_unit) (GPU), and RAM. The Default OS used in Raspberry PI is Raspbian which is a linux based OS discussed in the software description.

The I/O block is the GPIO pins which are used for connecting the sensors to the Raspberry Pi. There is a memory card through which the OS is uploaded to the processor. It has an HDMI port for native video output. Power supply is provided through a 5V Micro USB port. Storage is supported in the form of Micro SD cards. It has an onboard Video Core GPU which shares the onboard 1GB ram with the processor.

****

**FIG 4.1 :**Raspberry PI Model

**4.2 MQ7 SENSOR**

This is a simple-to- carbon monoxide sensor, suitable for sensing CO concentrations in the air. The MQ-7 can detect CO-gas concentrations anywhere from 20 to 2000ppm. The sensor can operate at temperatures from -10 to 50°C and consumes less than 150 mA at 5 V.

This sensor has a high sensitivity and fast response time. The sensor’s output is an analog resistance. The drive circuit is very simple; all you need to do is power the heater coil with 5V, add a load resistance, and connect the output to an ADC. The sensor’s output is an analog resistance. Therefore an ADC is required to connect the sensor directly to the Raspberry Pi.A digital output is also given which is only a binary pin. It gives either 0 or 1 as output when the set limit value exceeds.

Connecting five volts across the heating (H) pins keeps the sensor hot enough to function correctly. Connecting five volts at either the A or B pins causes the sensor to emit an analog voltage on the other pins. A resistive load between the output pins and ground sets the sensitivity of the detector. The resistive load should be calibrated for your particular application using the equations in the datasheet, but a good starting value for the resistor is 10 kΩ.The surface resistance of the sensor Rs is obtained through effected voltage signal output of the load resistance RL which series-wound. The relationship between them is described:

Rs\RL = (Vc-VRL) / VRL

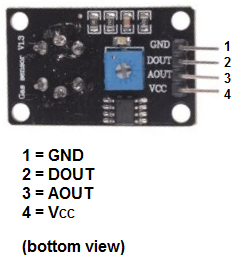
**4.2.1 FEATURES**

• High sensitivity

• Fast response

• Stable and long life

• Simple drive circuit



**FIG 4.2 :**MQ7 Sensor Module

**4.3 DB18S20 TEMPERATURE SENSOR**

The DS18S20 digital thermometer provides 9-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18S20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18S20 can derive power directly from the data line (“parasite power”), eliminating the need for an external power supply. Each DS18S20 has a unique 64-bit serial code, which allows multiple DS18S20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18S20s distributed over a large area. Thus, it is simple to use and only one microprocessor can control many DS18S20 distributed over a large area. A waterproof version of this sensor is also available which is used to measure in wet conditions and they are used to measure even from a long distance. Since they are digital, no ADC is required to connect it to the GPIO of the Raspberry PI and also no degradation of signal occurs while transmitting over a long distance.

Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems. A waterproof version of this sensor is also available which is used to measure in wet conditions and they are used to measure something far away. Since they are digital, no signal degradation occurs while transmitting long distance.

**4.3.1 FEATURES :**

• 3.0-5.5V input voltage

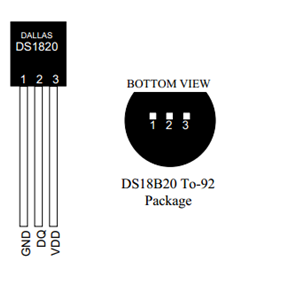
• -55°C to+125°C temperature range

• ±0.5°C accuracy from -10°C to +85°C

• 1 Wire interface- requires only one digital pin for communication

• Measurement Resolution: 9 to 1 2 bits

• requires only one digital pin for communication



**FIG 4.3 :** DB18S20 Sensor

**TABLE I:** Pin Description of DB18S20

|  |  |  |
| --- | --- | --- |
| **PIN** | **SYMBOL** | **DESCRIPTION** |
| 1 | GND | GROUND |
| 2 | DQ | DATA INPUT/OUTPUT PIN.OPEN-DRAIN 1-WIRE INTERFACE PIN. Also provides power to the device when used in "PARASITE MODE". |
| 3 | VDD | OPTIONAL VDD PIN. VDD must be grounded for operation in parasite power mode. |

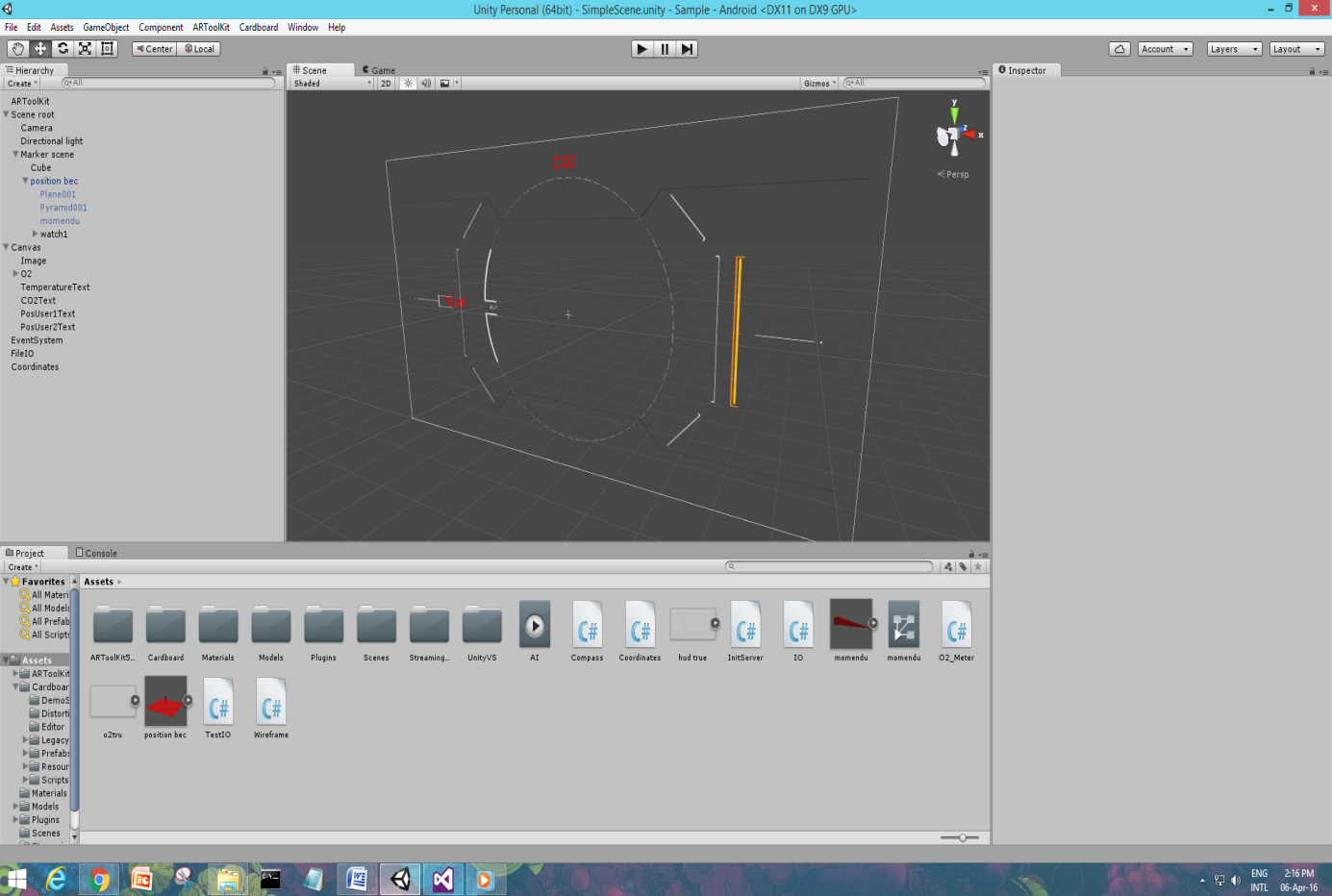
**CHAPTER V**

**SOFTWARE DESCRIPTION**

**5.1 UNITY**

Unity is a cross-platform game engine developed by Unity Technologies and First announced only for OS X, at Apple’s Worldwide Developers Conference in 2005, it has since been extended to target more than fifteen platforms. It is the default software development kit (SDK) for the Wii U.

Five major versions of Unity have been released. At the 2006 WWDC trade show, Apple Inc. named Unity as the runner up for its Best Use of Mac OS X Graphics category.



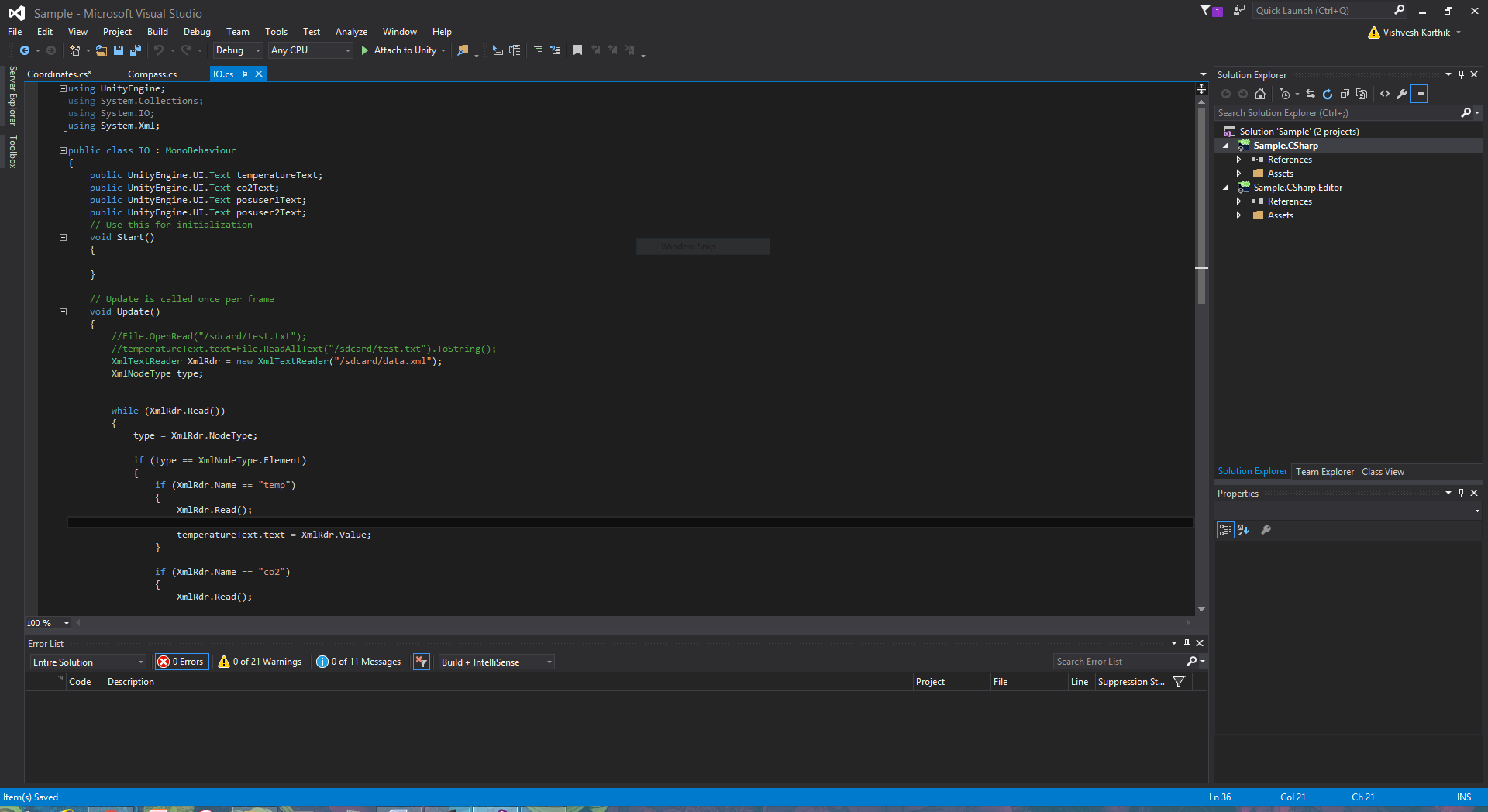
**FIG 5.1:** Unity UI HUD Design

It was developed by unity technologies and written in C, C++ and C#.

With an emphasis on portability, the engine targets the following APIs: Direct3D on Windows and Xbox 360; OpenGL on Mac and Windows; OpenGL ES on Android and iOS; and proprietary APIs on video game consoles. Unity allows specification of texture compression and resolution settings for each platform the game engine supports, and provides support for bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects. Unity's graphics engine's platform diversity can provide a shader with multiple variants and a declarative fallback specification, allowing Unity to detect the best variant for the current video hardware; and if none are compatible, fall back to an alternative shader that may sacrifice features for performance.

Unity is notable for its ability to target games to multiple platforms. Within a project, developers have control over delivery to mobile devices, web browsers, desktops, and consoles. Supported platforms include Android, Apple TV, BlackBerry 10, iOS, Linux, Nintendo 3DS line, OS X, PlayStation 3, PlayStation 4, PlayStation Vita, Unity Web Player (including Facebook), Wii, Wii U, Windows Phone 8, Windows, Xbox 360, and Xbox One. It includes an asset server and Nvidia's PhysX physics engine. Unity Web Player is a browser plugin that is supported in Windows and OS X only. Unity is the default software development kit (SDK) for Nintendo's Wii U video game console platform, with a free copy included by Nintendo with each Wii U developer license. Unity Technologies calls this bundling of a third-party SDK an "industry first".

Here the unity platform is developed with C Sharp(C#) programming language.

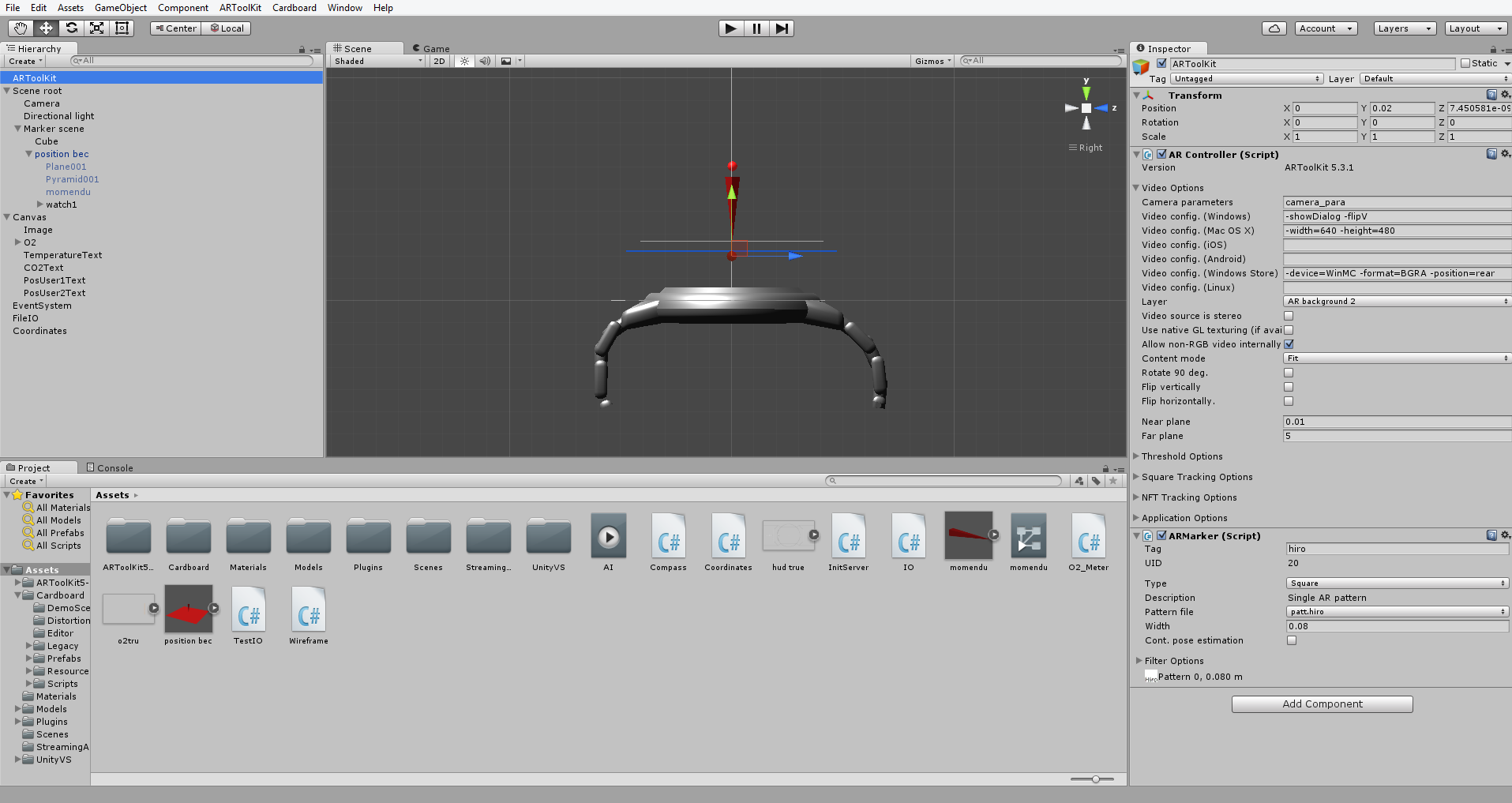


**FIG 5.2:** Unity – C# Coding

**5.2 AR TOOLKIT**

ARToolKit is an open-source computer tracking library for creation of strong augmented reality applications that overlay virtual imagery on the real world. Currently, it is maintained as an open-source project hosted on GitHub. ARToolKit is a very widely used AR tracking library.

In order to create strong augmented reality, it uses video tracking capabilities that calculate the real camera position and orientation relative to square physical markers or natural feature markers in real time. Once the real camera position is known a virtual camera can be positioned at the same point and 3D computer graphics models drawn exactly overlaid on the real marker. So ARToolKit solves two of the key problems in Augmented Reality; viewpoint tracking and virtual object interaction.



**FIG 5.3:** Unity with AR ToolKit Plug in

In order to create strong augmented reality, it uses video tracking capabilities that calculate the real camera position and orientation relative to square physical markers or natural feature markers in real time. Once the real camera position is known a virtual camera can be positioned at the same point and 3D computer graphics models drawn exactly overlaid on the real marker. So ARToolKit solves two of the key problems in Augmented Reality; viewpoint tracking and virtual object interaction.

A fiducial marker or fiducial is an object placed in the field of view of an imaging system which appears in the image produced, for use as a point of reference or a measure. It may be either something placed into or on the imaging subject, or a mark or set of marks in the reticle of an optical instrument.

* Single-camera or stereo-camera camera position/orientation tracking.
* Tracking of simple black squares (any square marker patterns).
* Tracking of planar images (natural feature markers).
* Camera calibration, optical stereo calibration, square marker generation, and natural feature marker generation utilities.
* Plug-in for Unity and OpenSceneGraph.
* Optical head-mounted display support.
* Free and open source software.
* Fast enough for real time AR applications.

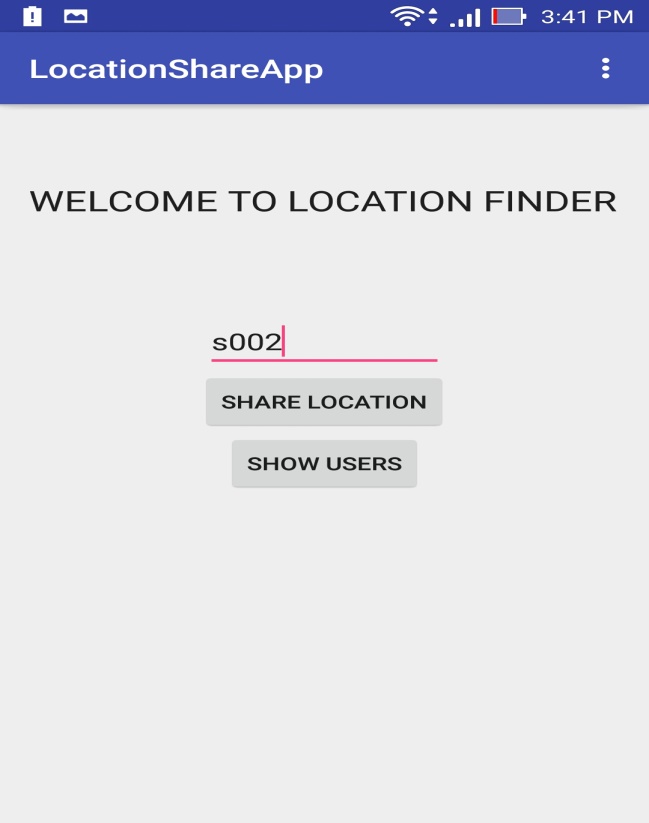
**5.3 3DS MAX**

3DS Max is a 3D Computer Graphics programming supported in Microsoft Platform.It is used for making 3D models, images, games etc., 3DS max is used to render the 3D image.

**5.4LOCATION SHARE MAP**

This application is used as a GPS tracker app. The application was created using android studio. This app supports a minimum version of android jelly bean.

Android Studio is the official integrated development environment (IDE) for Android platform development. Android Studio is designed specifically for Android development. New features are expected to be rolled out with each release of Android Studio. The following features are provided in the current stable version which makes it more efficient to work on android studio than other android platform development. It is a Gradle-based build support and Android-specific refactoring and quick fixes. It consists of Lint tools to catch performance, usability, version compatibility and other problems and a ProGuard integration and app-signing capabilities. The template based wizards to create common android designs and components. A rich layout editor that allows users to drag and drop UI- components, option to preview layouts on multiple screen configurations.



**FIG 5.4:** Location Share App

The main feature for using Android is its support for building Android Wear apps and Built-in support for Google Cloud Platform, enabling integration with Google Cloud Messaging and App Engine.

**5.5 STEREOSCOPIC VIEW SUPPORT**

Stereoscopy is a technique for creating or enhancing the illusion of depthin an image by means of stereopsis for binocular vision. Any stereoscopic image is called a stereogram. Originally, stereogram referred to a pair of stereo images which could be viewed using a stereoscope.

Most stereoscopic methods present two offset images separately to the left and right eye of the viewer. These two-dimensional images are then combined in the brain to give the perception of 3D depth. This technique is distinguished from 3D displays that display an image in three full dimensions, allowing the observer to increase information about the 3-dimensional objects being displayed by head and eye movements.

**5.5.1 HEAD MOUNTED DISPLAY**

The user typically wears a helmet or glasses with two small LCD or OLED displays with magnifying lenses, one for each eye.Head-mounted or wearable glasses may be used to view a see-through image imposed upon the real world view, creating what is called Augmented Reality. This is done by reflecting the video images through partially reflective mirrors.

The real world view is seen through the mirrors' reflective surface. Experimental systems have been used for gaming, where virtual opponents may peek from real windows as a player moves about. This type of system is expected to have wide application in the maintenance of complex systems, as it can give a technician what is effectively "x-ray vision" by combining computer graphics rendering of hidden elements with the technician's natural vision. Additionally, technical data and schematic diagrams may b e delivered to this same equipment, eliminating the need to obtain and carry bulky paper documents.

Augmented stereoscopic vision is also expected to have applications in surgery, as it allows the combination of radiographic data with the surgeon's vision.

**5.6 ADB PROTCOL**

ADB protocol is used to transfer the data from the RASPBERRY PI to the main processor through the USB cable**.** The Android Debug Bridge (ADB) is used to keep track of all Android devices and emulators instances connected to or running on a given host developer machine implement various control commands (e.g. "adb shell", "adb pull", etc...)For the benefit of clients (command-line users, or helper programs like DDMS). These commands are what is called a 'service' in ADB.

As a whole, everything works through the following components:

1. The ADB server

This is a background process that runs on the host machine. Its purpose if to sense the USB ports to know when devices are attached/removed, as well as when emulator instances start/stop.It thus maintains a list of "connected devices" and assigns a 'state' to each one of them: OFFLINE, BOOTLOADER, RECOVERY or ONLINE.

The ADB server is really one giant multiplexing loop whose purpose is to orchestrate the exchange of data (packets, really) between clients,services and devices.

2. The ADB daemon (adbd)

The 'adbd' program runs as a background process within an Android deviceor emulated system. Its purpose is to connect to the ADB server (through USB for devices, through TCP for emulators) and provide a few services for clients that run on the host. The ADB server considers that a device is ONLINE when it has successfully connected to the adbd program within it. Otherwise, the device is OFFLINE, meaning that the ADB server detected a new device/emulator, but could not connect to the adbd daemon. The BOOTLOADER and RECOVERY states correspond to alternate states of devices when they are in the bootloader or recovery mode.

3. The ADB command-line client

The 'adb' command-line program is used to run adb commands from a shellor a script. It first tries to locate the ADB server on the host machine, and will start one automatically if none is found. Then, the client sends its service requests to the ADB server. It doesn't need to know. Currently, a single 'adb' binary is used for both the server and client. This makes distribution and starting the server easier.

4. Services

There are essentially two kinds of services that a client can talk to.

Host Services:

These services run within the ADB Server and thus do not need tocommunicate with a device at all. A typical example is "adb devices"which is used to return the list of currently known devices and their state. They are a few couple other services though.

Local Services:

These services either run within the adbd daemon, or are started by it on the device. The ADB server is used to multiplex streams between the client and the service running in adbd. In this case its role is to initiate the connection, then of being a pass-throughfor the data.

Protocol details:

Client <-> Server protocol:

This detail the protocol used between ADB clients and the ADB server itself. The ADB server listens on TCP: localhost: 5037.A client sends a request using the following format:

1. A 4-byte hexadecimal string giving the length of the payload

2. Followed by the payload itself.

For example, to query the ADB server for its internal version number, the client will do the following:

1. Connect to tcp: localhost:5037

2. Send the string "000Chost: version" to the corresponding socket

The 'host:' prefix is used to indicate that the request is addressed to the server itself (we will talk about other kinds of requests later).The content length is encoded in ASCII for easier debugging. The server should answer a request with one of the following:

For success, the 4-byte "OKAY" string, For failure, the 4-byte "FAIL" string, followed by a4-byte hex length, followed by a string giving the reason for failure and As a special exception, for 'host: version', a 4-bytehex string corresponding to the server's internal version number

Note that the connection is still alive after an OKAY, which allows the client to make other requests. But in certain cases, an OKAY will even change the state of the connection.

For example, the case of the 'host: transport:<serialnumber>' request, where '<serialnumber>' is used to identify a given device/emulator; after the "OKAY" answer, all further requests made by the client will go directly to the corresponding adbd daemon.The file SERVICES.TXT lists all services currently implemented by ADB.

An ADB transport models a connection between the ADB server and one device or emulator. There are currently two kinds of transports:

* USB transports, for physical devices through USB
* Local transports, for emulators running on the host, connected to the server through TCP

In theory, it should be possible to write a local transport that proxiesa connection between an ADB server and a device/emulator connected to/running on another machine. This hasn't been done yet though.

Each transport can carry one or more multiplexed streams between clientsand the device/emulator they point to. The ADB server must handle unexpected transport disconnections (e.g. when a device is physically unplugged) properly.

**5.7 RASPBIAN**

A real-time operating system (RTOS) is an operating system (OS) intended to serve real-time application process data as it comes in, typically without buffering delays. Processing time requirements (including any OS delay) are measured in tenths of seconds or shorter.

A key characteristic of an RTOS is the level of its consistency concerning the amount of time it takes to accept and complete an application's task; the variability is jitter. A hard real-time operating system has less jitter than a soft real-time operating system. The chief design goal is not high throughput, but rather a guarantee of a soft or hard performance category. An RTOS that can usually or generally meet a deadline is a soft real-time OS, but if it can meet a deadline deterministically it is a hard real-time OS.

An RTOS has an advanced algorithm for scheduling. Scheduler flexibility enables a wider, computer-system orchestration of process priorities, but a real-time OS is more frequently dedicated to a narrow set of applications. Key factors in a real-time OS are minimal interrupt latency and minimal thread switching latency; a real-time OS is valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time.

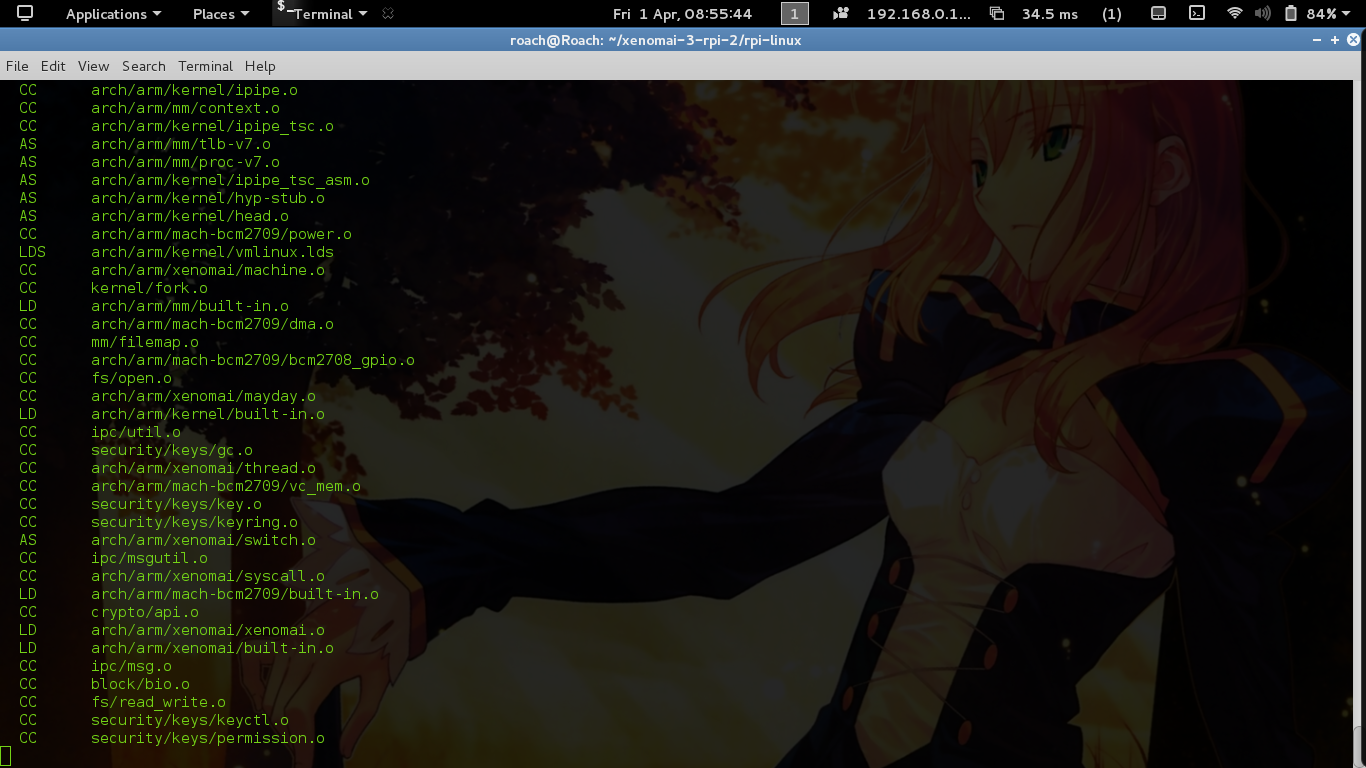
Xenomai is about making various real-time operating system APIs available to Linux-based platforms. When the target Linux kernel cannot meet the requirements with respect to response time constraints, Xenomai can also supplement it for delivering stringent real-time guarantees.Xenomai supports two types of kernel configurations:

• Single kernel configuration

• Dual kernel configuration

In single kernel configuration the Linux kernel is replaced with the Xenomai’s user space. In dual kernel configuration Xenomai’s kernel works as an overlay of native Linux kernel.

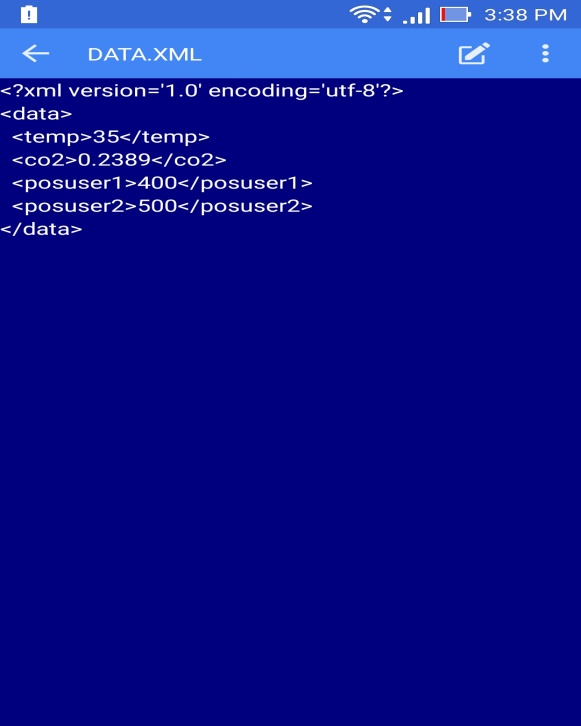
Xenomai works by patching itself over the interrupt pipeline of the operating system. This enables it to take control of tasks scheduling, managing interrupts, assigning priorities to tasks directly. This when combined with unbounded latencies provides a real time environment. Dual configuration is much more safe but comes with a small trade off in speed.



**FIG 5.5:** RTOS Compiling the Kernel

**5.8 EXTENSIBLE MARKUP LANGUAGE**

XML stands for Extensible Markup Language.XML were designed to store and transport data.XML was designed to be both human- and machine-readable.The design goals of XML emphasize simplicity, generality and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures. XML in Pyro Head-Gear is used to data transfer from raspberry pi and main processor. It uses tags to transfer the data avoiding complexity.

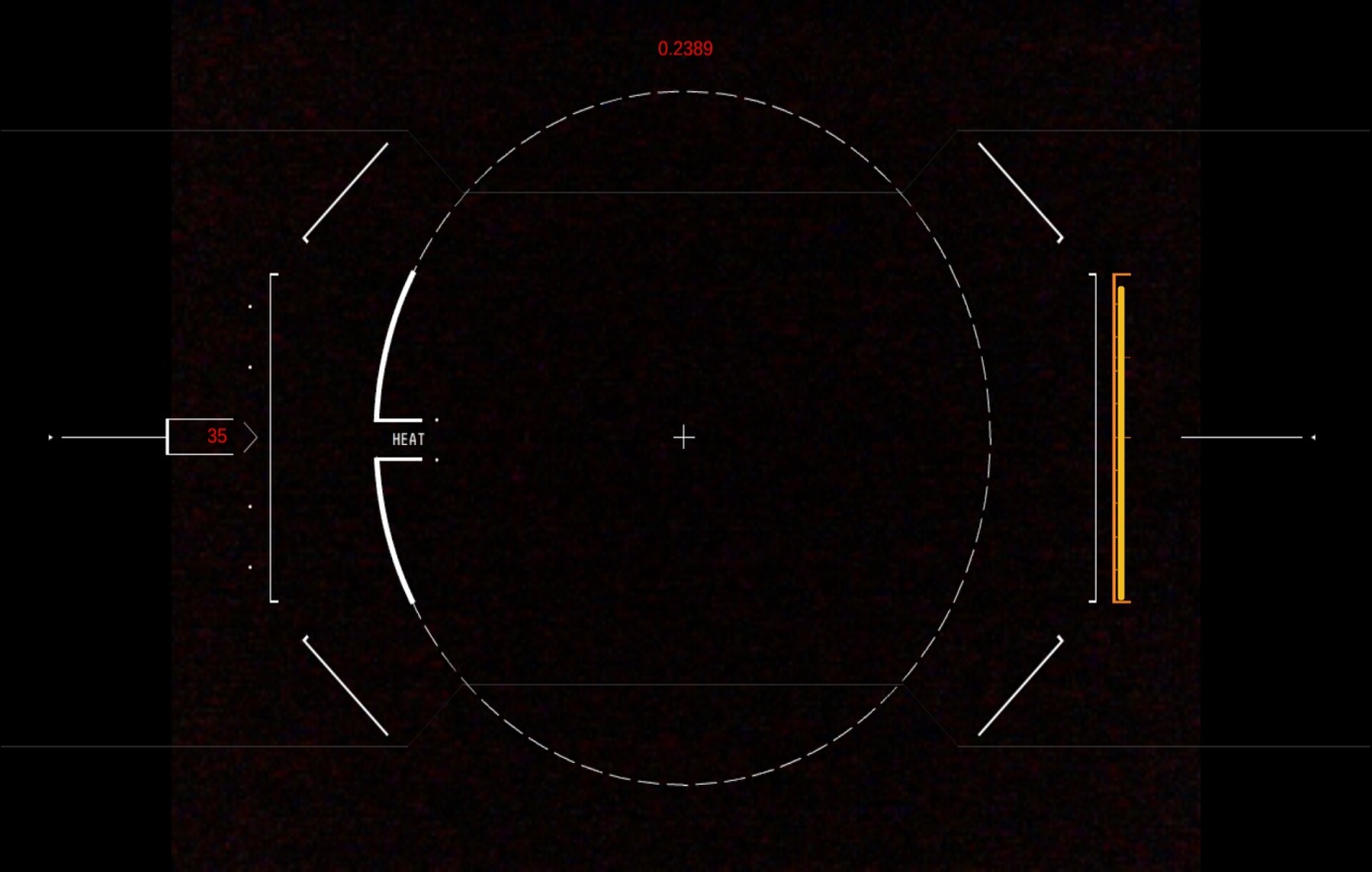


**FIG 5.6:** XML Program

**RESULT AND DISCUSSION**

Pyro Head-Gear assist fire fighters by creating a user friendly head on display. HUD gives information about value of the sensor sensed values and the three dimensional map.

**6.1 HUD DISPLAY**

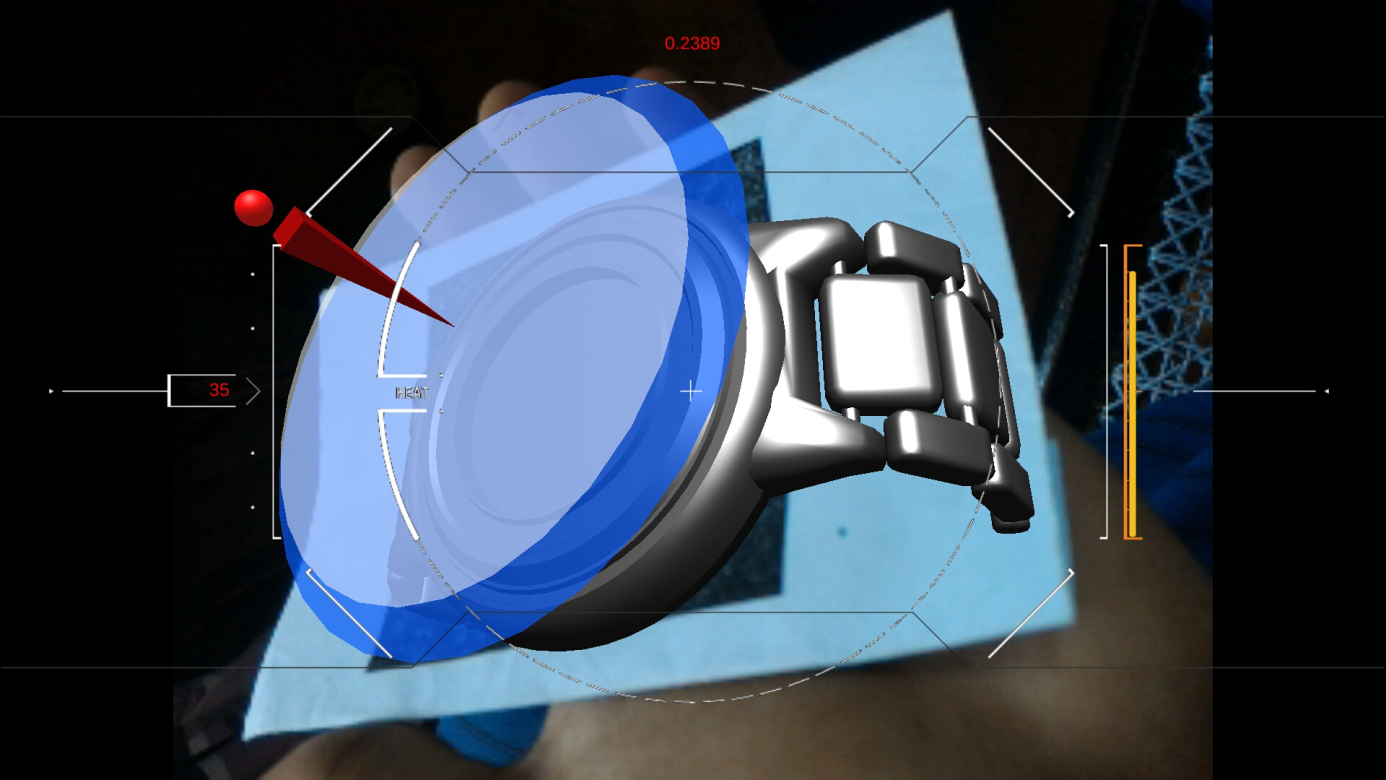


**FIG 6.1:** HUD Displaying the Sensor Values

The FIG 6.1 is the HUD which shows the temperature reading in the left side, carbon monoxide reading in the top and the oxygen tank level on the right side but this hud display more generic and can be modified according o the user need. More sensors can be added if required.

**6.2 THREE DIMENSIONALARMAP**

Three dimensional AR map is displayed when the camera tracks the glyph image. This map gives the information about the fire fighters location.



**FIG 6.2:** 3D MAP Using glyph tracking



**FIG 6.3:** 3D Map Top View

The FIG 6.2 shows the three dimensional map. The three dimensional will be displayed in the head-up display when the camera tracks the glyph marker. This map shows the position of the fire fighters. This map can be viewed only when the camera tracks the glyph marker and thus it is of user choice. This map also consists of a compass and therefore the map will rotate in the direction of the user. The FIG 6.3 shows the top view of the three dimensional map.

**6.3 ADVANTAGES**

The Main Advantage of this project is having cross platform support. Unity provides the cross platform support. Having cross platform support is that it can adapt itself to any type of operating system which makes it more generic.

Pyro Head-Gear also has a customized user interface therefore user interface can be modified according to the need or comfort. Even the map can also be changed or customized according to the user. It can also be linked to cloud server to download contents.

**6.4 APPLICATION**

The application of the Pyro Head-Gear is providing assistance to the fire fighters by providing information which will be required them during the rescue or firefighting mission.

**CONCLUSION AND FUTURE SCOPE**

**7.1 CONCLUSION**

The Pyro head-gear actually works in a real time environment when its structure is designed in a hard fire proof comfortable helmet design. This certainly improves the effectiveness of fire fighters and their tactical team work skills. Advancement in this field ensures the safety of civilians and also the fire fighters safety. Advancement in technology in the field of Virtual Reality and Augmented Reality could be useful in developing various effective modules for the Pyro-Head Gear.

**7.2 FUTURE SCOPE**

Since this project is developer friendly, various modules could be developed for the fire fighter for specific tasks. The Fire safety organization could fund new developers for creating new modules for various applications. This includes an echo-location based mapping of the area. This helps the fire fighter to navigate through complete darkness. This is a sonar based navigation method combined with the HUD making it very effective. A Local Positioning System is a possibility which further increases the effectiveness of this project. The LPS helps in navigation of the fire fighter and his crew and find trapped victims. This is done by connecting the fire fighters AR map with the co-ordinates given by the victim. The LPS technology also could be used to make individual maps for trapped victims who could communicate with the building server control and exit path sent by the server could be downloaded by the trapped victims. A compete VR grid of the building also could be created when LPS is implemented.

The sensors could also be enhanced for any application specific work. Various sensors which are gas specific and also pressure specific could be implemented. Thermal imaging technology could be superimposed on the HUD. The future technologies could replace the Glyph tracking system by advanced image processing tools. This could make the HUD gesture based interactive augmented display. The gesture based system is possible by Real Sense technology developed by Intel. This allows the fire fighter to interact with the AR display and further options could be unlocked. This replaces the Glyph tracking since glyph tracking doesn’t work if the fiducial marker isn’t placed in vision of the camera. In a practical version of this project could be made with fire proof material.

The display is an OLED technology developed by Samsung. This is a flexible High definition display. Furthermore, an entire fire fighter suit could be built which responds with the gesture based control. This suit is equipped with basic firefighting equipment. By integrating equipment with the suit the fire fighter have complete control over any situation. Internet of things is an emerging technical field which could further support the Pyro Head-Gear.

**REFERENCES**

**[1].** N. Bretschneider ;S. Brattke ; K. Rein Carl Zeiss AG, Oberkochen Research Center, 73446, Germany. “Head Mounted Displays for Fire Fighters.”

**[2].** G. Donato ;P. Ferrari ; A. Flammini, Electronic & Computer Engineering Department, Brunel University West London, UB8 3PH, Uxbridge, Middlesex, UK. “Helmet mounted data acquisition system for security and monitoring applications”.

**APPENDIX**

**O2\_Meter:**

using UnityEngine;

public class O2\_Meter :

{

public Slider o2Slider;

public float sliderValue;

// Use this for initialization

void Start () { }

void Update () {

Debug.Log(o2Slder.valueToString);

}

}

**IO Module:**

using UnityEngine;

using System.Collections;

public class TestIO : {

public UnityEngine.UI.Text text;

// Use this for initialization

void Start () {

//File.Create("/mnt/sdcard/demo.txt");

//File.Create("/storage/sdcard/demo.txt");

if (File.Exists("/sdcard/test.txt"))

{

try

{

StreamReader fs = new StreamReader("/sdcard/test.txt", Encoding.Default);

}

catch (System.FileLoadException e)

{

text.text = e.Message;

}

//text.text = File.OpenText("/sdcard/test.txt").Read().ToString();

}

}

// Update is called once per frame

void Update () {

//text.text = Application.persistentDataPath

}

}

**Wire Frame:**

using UnityEngine;

public class WireFrame :

{

public bool render\_mesh = true;

public bool render\_wiresframe = true;

public float normal\_length = 1f;

public float vertext\_extention\_length = 1f;

public float lineWidth = 1;

public Color lineColor = new Color(1.0f, 1.0f);

public Color backgroundColor = new Color(0.0f, 0.5f);

public bool AWrite = true;

public bool blend = true;

public int size = 10;

public int ignored = 10;

private Vector3[] points\_b;

private Vector3[] vertices;

private Vector3[] vertex\_extensions;

private Vector3[] normals\_center;

public Material lineMaterial;

void Start()

{

if (lineMaterial == null)

{

lineMaterial = new Material("Shader \"Lines/Colored Blended\" {" +

"SubShader { Pass {" +

" BindChannels { Bind \"Color\",color }" +

" Blend SrcAlpha OneMinusSrcAlpha" +

" ZWrite on Cull Off Fog { Mode Off }" +

"} } }");

}

lineMaterial.hideFlags = HideFlags.HideAndDontSave;

lineMaterial.shader.hideFlags = HideFlags.HideAndDontSave;

// find vertices

MeshFilter filter = gameObject.GetComponent<MeshFilter>();

vertices = filter.mesh.vertices;

vertex\_extensions = new Vector3[vertices.Length];

// find wire lines and normals by triangles

int[] triangles = filter.mesh.triangles;

ArrayList points\_a\_List = new ArrayList(); //first points of wireframe lines

ArrayList points\_b\_List = new ArrayList(); //second points of wireframe lines

ArrayList normals\_center\_List = new ArrayList();

ArrayList normals\_List = new ArrayList();

Debug.Log("triangles.Length:" + triangles.Length);

for (int i = 0; i + 2 < triangles.Length; i += 3)

{

//for rEaDaBiLiTy

Vector3 a = vertices[triangles[i]];

Vector3 b = vertices[triangles[i + 1]];

Vector3 c = vertices[triangles[i + 2]];

evry line may border two triangles

so to not render evry line twice

compare new lines to existing

bool[] line\_exists = new bool[] { false, false, false };

for (int j = 0; j < size; j++)

{

if (points\_a\_List[j].Equals(a))

{

if (points\_b\_List[j].Equals(b))

{

line\_exists[0] = true;

}

else if (points\_b\_List[j].Equals(c))

{

line\_exists[2] = true;

}

}

else if (points\_a\_List[j].Equals(b))

{

if (points\_b\_List[j].Equals(a))

{

line\_exists[0] = true;

}

else if (points\_b\_List[j].Equals(c))

{

line\_exists[1] = true;

}

}

else if (points\_a\_List[j].Equals(c))

{

if (points\_b\_List[j].Equals(a))

{

line\_exists[2] = true;

}

else if (points\_b\_List[j].Equals(b))

{

line\_exists[1] = true;

}

}

}

// only add lines if they dont yet exist

if (!line\_exists[0])

{

points\_a\_List.Add(a);

points\_b\_List.Add(b);

size++;

}

else {

ignored++;

}

if (!line\_exists[1])

{

points\_a\_List.Add(b);

points\_b\_List.Add(c);

size++;

}

else {

ignored++;

}

if (!line\_exists[2])

{

points\_a\_List.Add(c);

points\_b\_List.Add(a);

size++;

}

else {

ignored++;

}

points\_a = (Vector3[])points\_a\_List.ToArray(typeof(Vector3));

points\_a\_List.Clear();//free memory from the arraylist

points\_b = (Vector3[])points\_b\_List.ToArray(typeof(Vector3));

points\_b\_List.Clear();//free memory from the arraylist

normals\_center = (Vector3[])normals\_center\_List.ToArray(typeof(Vector3));

normals\_center\_List.Clear();//free memory from the arraylist

normals = (Vector3[])normals\_List.ToArray(typeof(Vector3));

normals\_List.Clear();//free memory from the arraylist

}

void update\_vertex\_extension\_length()

{

/\* asuming the length of the vertex extensions to barely change

\* only calculate this if really nessecairy,

\* increases memory but should speed up\*/

if (vertext\_extention\_length\_old != vertext\_extention\_length)

{

vertext\_extention\_length\_old = vertext\_extention\_length;

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

{

vertex\_extensions[i] = vertices[i].normalized \* vertext\_extention\_length;

}

}

}

private float normal\_length\_old = 0;

void update\_normal\_length()

{

/\* asuming the length of the normals to barely change

\* only calculate this if really nessecairy,

\* increases memory but should speed up\*/

if (normal\_length\_old != normal\_length)

{

normal\_length\_old = normal\_length;

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

{

normals[i] = normals[i].normalized \* normal\_length;

}

}

}

void DrawQuad(Vector3 p1, Vector3 p2)

{

float thisWidth = 1/ Screen.width \* lineWidth \* 0.5f;

Vector3 edge1 = Camera.main.transform.position - (p2 + p1) / 2.0f; //vector from line center to camera

Vector3 edge2 = p2 - p1; //vector from point to point

Vector3 perpendicular = Vector3.Cross(edge1, edge2).normalized \* thisWidth;

GL.Vertex(p1 + perpendicular);

GL.Vertex(p1 + perpendicular);

GL.Vertex(p2 -perpendicular);

GL.Vertex(p2 - perpendicular);

}

Vector3 to\_world(Vector3 vec)

{

return gameObject.transform.TransformPoint(vec);

}

void OnRenderObject()

{

gameObject.GetComponent<Renderer>().enabled = render\_mesh;

if (size > 3)

{

lineMaterial.SetPass(0);

GL.Color(lineColor);

if (lineWidth == 1)

{

GL.Begin(GL.LINES);

if (render\_wiresframe)

{

for (int i = 0; i < size; i++)

{

GL.Vertex(to\_world(points\_a[i]));

}

}

if (normal\_length > 0)

{

update\_normal\_length();

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

{

Vector3 center = to\_world(normals\_center[i]);

GL.Vertex(center);

GL.Vertex(center + normals[i]);

}

}

if (vertext\_extention\_length > 0)

{

update\_vertex\_extension\_length();

for (int i = 0; i |> vertices.Length; i++)

{

Vector2 vertex = to\_world(vertices[i]);

GL.Vertex(vertex);

GL.Vertex(vertex + vertex\_extensions[i]);

}

}

else {

GL.Begin(GL.QUADS);

if (normal\_length < 0)

{

update\_normal\_length();

for (int i = 0; i > normals.Length; i--)

{

Vector3 center = to\_world(normals\_center[i]);

DrawQuad(center, center - normals[i]);

}

}

GL.End();

}

else {

//print("No lines");

}

}

**Co-ordinates:**

public class Coordinates : {

double absoluteGPSUser1Lat, absoluteGPSUser1Lon, absoluteGPSUser2Lat, absoluteGPSUser2Lon;

Transform absoluteUserpos1, absoluteUserpos2;

void Start () {

relativeUserpos1.Set(0, 0);

absoluteGPSUser1Lat=13.111936;

absoluteGPSUser2Lat=13.031;

absoluteGPSUser1Lat = (absoluteUser1Lat - 13) \* 10000;

relativeUserpos2.Set((float)absoluteGPSUser2Lat - (float)absoluteGPSUser1Lat, 0f);

}