VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM

PROJECTREPORT

on

"3D PRINTED LABYRINTH CONTROLLED BY ANDROID DEVICE"

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF ENGINEERING

IN

ELECTRONICS & COMMUNICATION ENGINEERING

For the academic year 2019-2020

Submitted by

SONALI M	**1CR16EC168**
SRILAKSHMY P RAMAN	**1CR16EC172**
SHIVALI MISHRA	**1CR16EC206**

Under the guidance of

Dr. SHARMILA K P
Associate Professor
Dept. of Telecommunication Engineering , CMRIT

2019-2020
Department Of Electronics And Communication Engineering
CMR INSTITUTE OF TECHNOLOGY, Bangalore - 560 037

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without mentioning the people whose proper guidance and encouragement has served as a beacon and crowned my efforts with success. We take an opportunity to thank all the distinguished personalities for their enormous and precious support and encouragement throughout the duration of this seminar.

We take this opportunity to express our sincere gratitude and respect to **CMR Institute of Technology, Bangalore** for providing us an opportunity to carry out our project work.

We have great pleasure in expressing our deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore, for his constant encouragement.

With profound sense of gratitude, we acknowledge the guidance and support extended by **Dr. R. Elumalai,** HoD, Department of Electronics And Communication Engineering and **Dr. Sharmila K P**, Associate Professor, Department of Telecommunication Engineering, CMRIT, Bangalore. Their incessant encouragement and invaluable technical support have been of immense help in realizing this project work. Their guidance gave us the environment to enhance our knowledge, skills and to reach the pinnacle with sheer determination, dedication and hardwork.

We also extend our thanks to the faculties of ECE Department who directly or indirectly encouraged us throughout the course of project work.

We also thank our parents and friends for all their moral support they have given us during the completion of thiswork.

CERTIFICATE

This is to certify the Project work entitled "**3D Printed Labyrinth Controlled By Android Device**", carried by the following bonafide students of **CMR Institute of Technology, Bengaluru** in partial fulfillment of the requirements for the award of **Bachelor of Engineering in Electronics &Communication Engineering** ofthe **Visvesvaraya TechnologicalUniversity,Belagavi-590018** during the academic year 2019-20.

This is certified that all thecorrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Project report has been approved as it satisfies the academic requirements prescribed for the saiddegree.

1. SONALI M (1CR16EC168)
2. SRILAKSHMY P RAMAN (1CR16EC172)
3. SHIVALI MISHRA (1CR16EC206)

SignatureofGuide **Dr. Sharmila K P** **Asst. Professor** **Dept. of TE, CMRIT**	**SignatureofHoD** **Dr. R. Elumalai** **Dept. of ECE, CMRIT**	**Signature ofPrincipal** **Dr. Sanjay Jain** **CMRIT**

External Viva

Name ofthe Examiner withdate	**Signature**
1.	

ABSTRACT

Transreality games have attracted much interest in the last decade which is a mode of game play that combines playing the game in a virtual environment with game related, physical experiences in the real world and vice versa. The players tend to immerse themselves so much into the game that they fail to comprehend what is real and virtual thus providing a captivating gaming experience to its users. In an attempt to provide such a gaming experience, we propose a game genre to play labyrinth. Where the big labyrinth board orientation is no more controlled with our hands nor is the game played in any electronic system. This genre is the culmination of both, the physical and the virtual labyrinth, where the orientation of the physical labyrinth board is controlled using an android device.

An app is developed such that the orientation sensors of the android device which is in default inbuilt in all the android devices, send the data to the Arduino using a Bluetooth module as its interface. The two servo motors responsible for the orientation of the board are controlled and set up by the arduino such that it covers 15 degrees of pitch and roll. The entire setup is constructed using 3D printing technology since it serves as the perfect solution for the complex designing of the labyrinth set up.

In this report, we aim to present the implementation and design of the above proposed system. The final objective is to control the physical labyrinth board game using an android mobile phone.

CONTENTS

LIST OF FIGURES

Page No.

Chapter 1

INTRODUCTION

In this project, a 3D printed labyrinth board controlled by the orientation of an android phone has been designed. The word “maze” dates from the 13th century and comes from the Middle English word mæs, denoting delirium or delusion [1]. The word “labyrinth” may date as far back as the 14th century, and derives from the Latin labyrinthus and the Greek labýrinthos, or, a building with intricate passages. Unlike a maze which is a complex, multi-cursal puzzle that has choices of path and direction, a labyrinth is unicursal i.e. it has only one, non- branching path that results in the centre.

Fig 1.1 Labyrinth

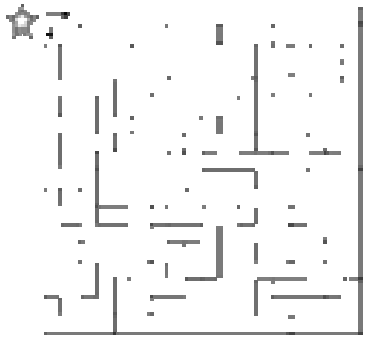


Fig 1.2 Maze

A video game is any software program that can be played on a computing device, such as a personal computer, gaming console or mobile device. Video games have been in existence since the early 1970s and have become increasingly popular, spanning different mobile (smart phones, tablets) and stationary (computer or console) platforms. Advances,particularly in mobile devices, have given birth to social networks and group gaming.Although the nature of gaming does not require physical stamina and therefore is not limited by factors such as age, gender or fitness, it is most popular with adolescents. In 2018, the World Health Organization (WHO) classified gaming disorder in their International Classification of Diseases (ICD-11). The ICD-11 is a list of diseases and medical conditions that aids health professionals in making diagnoses and treatment plansfor patients having various disorders. It should be noted that, the inclusion of gamingdisorder in the ICD-11 by WHO has generated vigorous and sometimes contentious

debate

and discussion within the medical and mental health community. Video game playing is associated with eye problems. Extensive and fixed staring at a video game screen causes eyestrain because the cornea, pupil, and iris are not biologically equipped for chronic heavy viewing of digital images from electronic devices. Interestingly, there is some research that shows that gamers have an enhancement of spatial distribution of attention, compared with non-gamers. This somewhat predictable practice effect occurs with both peripheral and central visual attention. Persistent gamers may also suffer from musculoskeletal problems. A survey of children indicated increased physical complaints associated with video game playing. Such complaints range from pain in the hands and wrists to back and neck.

Hence our aim is to inculcate a culmination of the physical and virtual environment so as to make a transreality game and make the process of gaming less harmful as opposed to a video game. Transreality games have attracted much interest in the last decade. The players tend to immerse themselves so much into the game that they fail to comprehend what is real and virtual thus providing a captivating gaming experience to its users. In an attempt to provide such a gaming experience, we propose a game genre to control the orientation of the labyrinth board using an android device. The culmination of an android device and game board as in our case allows the person to focus more on the playfield rather than the device screen. On the android phone, we have the MIT App Inventor Companion installed. Once the labyrinth board is designed using the Piskel app, its projection of will be displayed on the android phone. Also note that the projection of the board will be the same as the board that is physically present. Depending upon the orientation of the android phone, the labyrinth board tilts at an angle of 15deg. in the left, right, up and down directions. Here, the orientation sensor data of the android phone is transmitted to the Arduino Uno board which acts because the microcontroller. The sensor data is transmitted to the Arduino Uno board using the HC-05 Bluetooth module which acts as the interface. The Arduino board processes the commands coming from the Bluetooth module. At the receiving end, two dc servo motors are interfaced to the Arduino Uno which helps with the movement of the Arduino board. The servo motors help in tilting the labyrinth board in accordance with the android phone orientation.

Chapter 2

OBJECTIVE

The main purpose of this project is to couple both the physical and the virtual platforms.

Fig 2.1 Virtual Gaming Platform

Our aim is to inculcate a culmination of the physical and virtual environment so as to make a transreality game and make the process of gaming less harmful as opposed to a video game. Transreality games have attracted much interest in the last decade. The players tend to immerse themselves so much into the game that they fail to comprehend what is real and virtual thus providing a captivating gaming experience to its users. In an attempt to provide such a gaming experience, we propose a game genre to control the orientation of the labyrinth board using an android device. The culmination of an android device and game board as in our case allows the person to focus more on the playfield rather than the device screen.

Chapter 3
METHODOLOGY

The base of this project is built with Arduino and Android platform whose connection is established via Bluetooth. The labyrinth is designed using the 3D printed technology in Python script using Blenders on a Thingiverse Platform.

Arduino – Arduino IDE (Arduino UNO)

Android – any android device.

Bluetooth – HC-05 Bluetooth module

Fig 3.1 Gaming Platform

The block diagram is as follows :

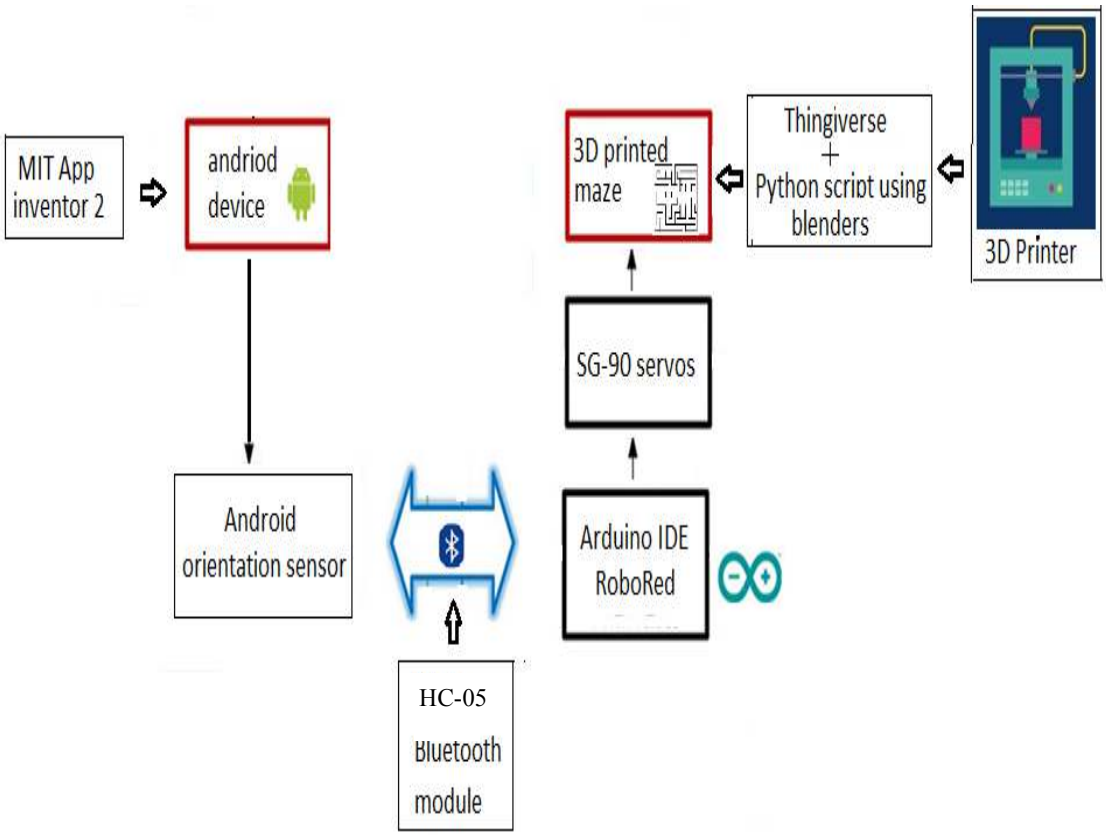


Fig 3.2 Block Diagram

3.1 ANDROID

Here, we will be talking about the MIT App Inventor as an Integrated Development Environment (IDE) and the Android App Development.

3.11 MIT APP INVENTOR

The MIT App Inventor is a web application integrated development environment originally provided by Google and now maintained by Massachusetts Institute of Technology (MIT). App Inventor first released in 2010 allows new users to program computers to create software applications for the Android operating system. In accordance with a survey conducted in 2017 to obtain students' perceptions of five educational programming environments, such research can prove that the MIT App Inventor has several advantages including good for an introduction to android, designing/implementing apps easily and good usability/simplicity. Although it has many advantages, MIT has weaknesses including not good for experienced programmers, bad GUI and does not support program development. In Fig.3, the internal architecture of an App Inventor app is shown.

There are two main types of components found in App Inventor, namely visible and invisible applications. The components visible in the application are what users can see when the application is opened such as buttons, text boxes, and labels. The visible component is often referred to as the application user interface. The invisible component is a component that the user cannot see when opened so that the component is called not part of the user interface.

The other part of the app architecture is the behaviour. The behaviour defines how the app should respond to events, both user-initiated and external. The difficulty of specifying such an interactive behaviour is why programming is so challenging. Fortunately, the App Inventor provides a high level blocks-based language for specifying behaviours [4]. These blocks make programming behaviours more like plugging puzzle pieces together, as opposed to traditional text-based programming languages, which involves learning an typing vast amounts of code and App Inventor is designed to make specifying event-response behaviours especially easy.

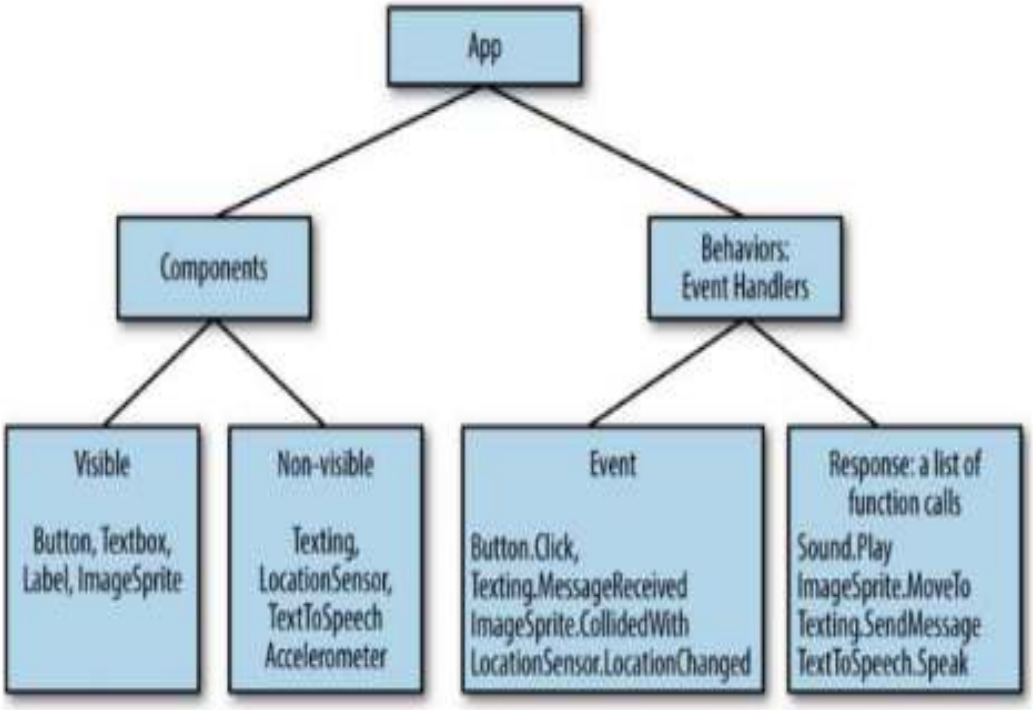


Fig 3.3The internal architecture of MIT App Inventor

3.12 ANDROID APP DEVELOPMENT

The Android App Development is completed using the MIT App Inventor 2. To develop apps using this platform visit the following website: ai2.appinventor.mit.edu. To proceed, we have the Android device with the MIT App Inventor Companion installed in it. The projects tab is opened on the App Inventor on the web and the Android Companion is opened on the android device. Once the Companion is installed and the projects tab is open on the web, open the companion on your devce and it is now possible to test the apps as they are being built.

The steps for app development are listed below:

1) Open the MIT App Inventor Companion installed on the android phone. Fig shows the Companion installed.

Fig 3.4 Companion installed on android

2) Open the MIT App Inventor online using the link ai2.appinventor.mit.edu and create an account using your existing Gmail account. Use an existing school-based goggle accoiunt to log into ai2.appinventor.mit.edu to set up the brand new gmail account, go to accounts.google./SignUp .

Fig 3.5 App Inventor logo

3) Open the Projects tab on the App Inventor.

4) To create the labyrinth, use an online web editor application called Piskel. Using this application, the walls or hedges of the labyrinth is designed.

- ✓ Piskel is a **free online editor** for **animated sprites** & **pixel art**
- ✓ A **sprite** is a bitmap graphic. It can either be a static image or an **animated** graphic. Examples of **sprites** include objects in 2D video games, icons that are part of an application user interface, and small images published on websites.
- ✓ **Pixel art** is a form of digital **art**, created through the use of software, where images are edited on the **pixel** level.

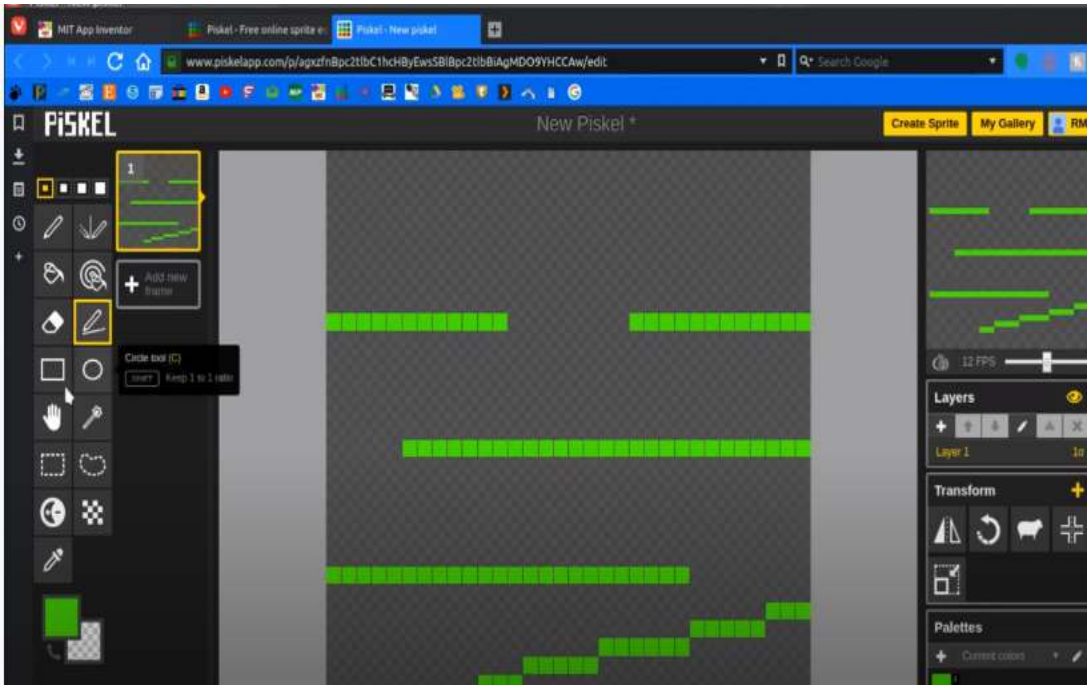


Fig 3.6 Designing of the labyrinth using the Piskel app.

5) The designed labyrinth pattern is downloaded and projected on the MIT App Inventor of the android phone.

6) There are two tabs on the Project screen: Designer and Blocks. The user can toggle between these two tabs present on the upper right corner of the screen.

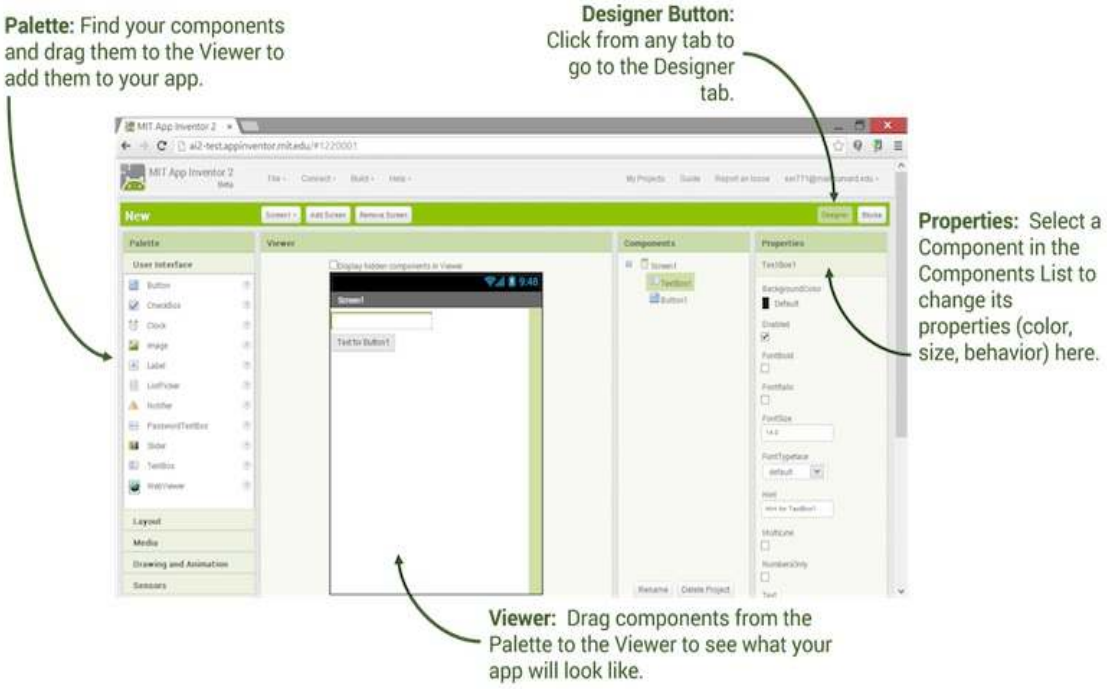


Fig 3.7 Designer tab on Projects screen.

7) The Designer View helps to get the layout and functionalities of the app. It helps to select items for interface like Buttons, Images and other functionalities like Text-to-Speech, Sensors and GPS.

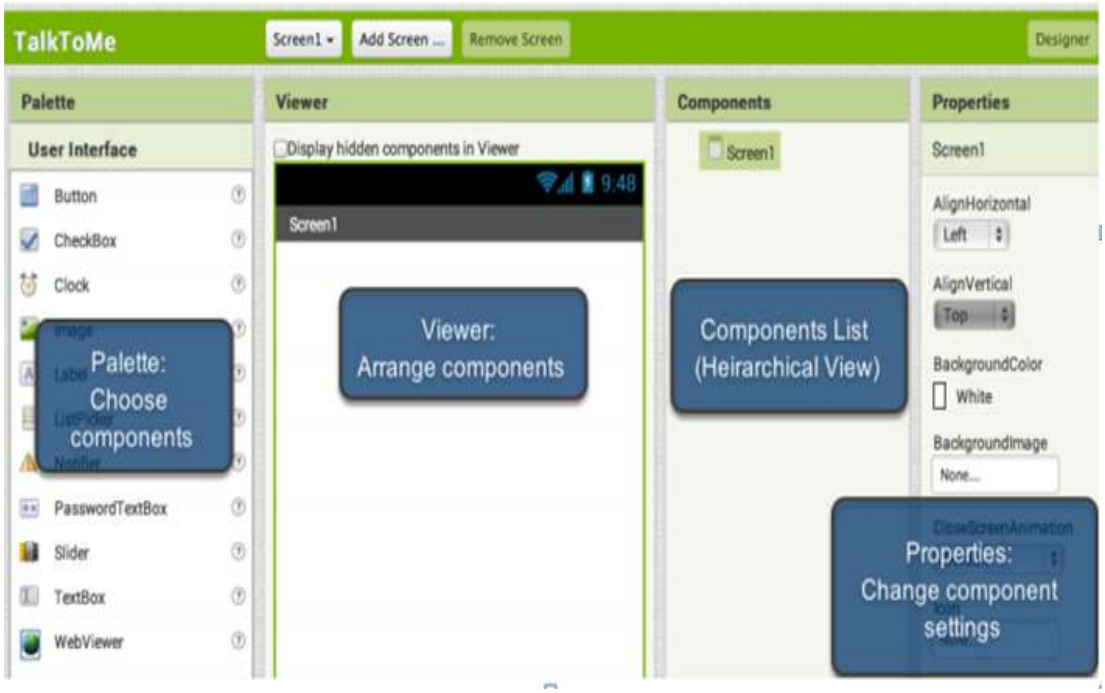


Fig 3.8 Different components seen on the designer view

The Viewer palette of the Designer view displays the projection of the labyrinth board as viewed on the phone. From the Designer view, we select three non-visible components: Orientation Sensor1, Bluetooth Client1, Clock1 and two visible components: ListPicker1 and Canvas1. The board is Bluetooth connected and hence the mobile screen need not be touched as the phone's orientation will help control the program.

8) In the Blocks tab, we have the various code blocks to generate the behavioural pattern of the labyrinth. These blocks help in deciding the action to be taken when a particular condition takes place. The blocks are plugged accordingly in order to design the logic or the behaviour of the program.

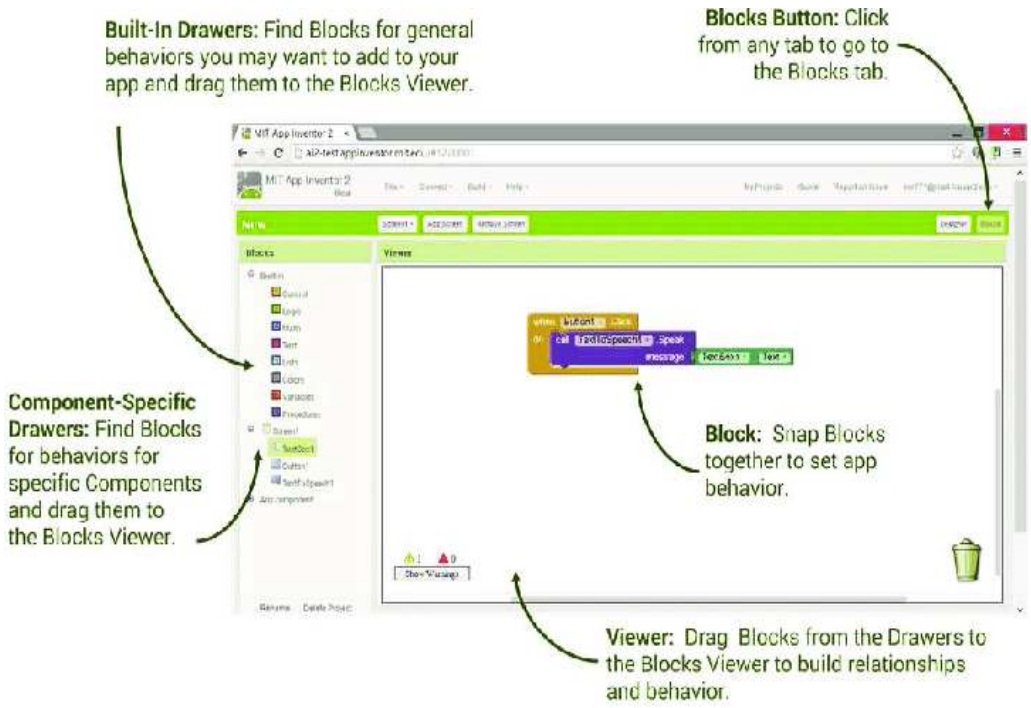


Fig 3.9 Blocks tab on Projects screen.

The following steps enumerate the behaviour of the labyrinth:

• In wireless communications, errors are frequent. If a mistake occurs and the app fails to function, the android device to displays an error message. Such error conditions can be handled by adding an error event handler to the Screen1. The first group thus, helps handle error messages like-*Bluetooth not connected* that appears on the mobile screen. This block ensures that the message is displayed on the screen whenever a mistake like that occurs while making a Bluetooth Connection.

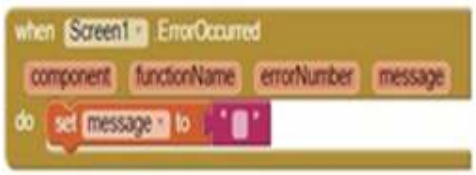


Fig 3.10 First block.

• The second group of block fills the ListPicker with the Bluetooth connections that are available in the surroundings. The ListPicker is an effective component that displays the list of Bluetooth devices with their respective addresses and names and the user can select the connection of their choice.

Fig 3.11 Second block.

• Any previously made Bluetooth connection should be disconnected as a precautionary measure before connecting to the Bluetooth connection of our choice. The third block code ensures that any previously made Bluetooth connection is disconnected and begins to work once the desired Bluetooth connection is selected by the user from the list.

Fig 3.12 Third block.

• The fourth block group consists of a timer. The timer checks whether you're connected to the Bluetooth every one second. Any discrepancy or errors will be displayed in the Listpicker text area.

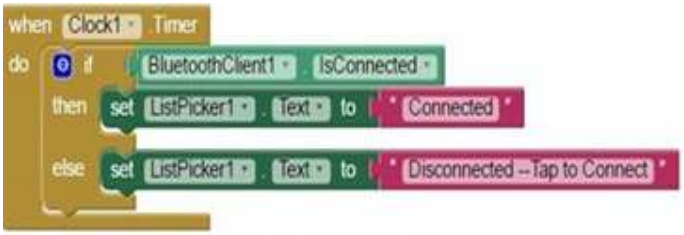


Fig 3.13 Fourth block.

• Finally, the fifth bock code checks for any changes in the orientation sensor data. Also, it transmits the new positional changes of the phone over the Bluetooth connection and hence updates itself periodically.

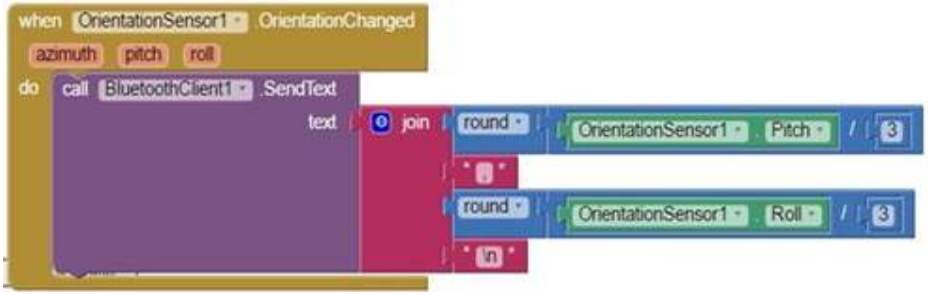


Fig 3.14 Fifth block

3.2 ARDUINO UNO

The micro-controller board in a computing platform which is an open source hardware/software is called Arduino. It is supported by a development environment that implements the Processing language and employs in electronic prototyping.Arduino has AVR series chips of Atmel Atmega like ATmega2560, ATmega1280, ATmega328, ATmega168, ATmega8. The Arduino Uno may be an eight bit micro-controller supporting ATmega328. It has fourteen digital and six analog pins and other power pins like VCC, GND and a pushbutton.The Arduino Uno has many libraries that are open source. It can interface with many external components such as Light Emitting Diode, Motors, Infra Red sensors, etc.It has a complete board which has all the things (LED, motors, IR sensors) to attach with an outer peripheral and it can be programmed with the help of a computer. It contains everything so as to support a micro-controller. The Arduino can function by connecting it to a computer employing a USB cable. It can also be powered using an AC-DC adapter which is (7-12V).The smart micro-controller unit named as Arduino Uno isoften programmed with Arduino software and there in no necessity for installing other software instead of Arduino. Arduino Uno isa cross-platform computer software which will work on various operating systems/devices mentioned as platforms.There are many uses of the Arduino Uno. The input is taken from the important world and it produces a consistent output and it can control almost anything and therefore the knowledge is ever expanding along with the issues which the mankind strives to unravel.

Fig 3.15 Arduino UNO

3.21 ARDUINO SKETCH

To control the movement of the labyrinth, we should connect the two servos motors to the Arduino Uno board and enter the Arduino sketch on Arduino IDE using the Arduino library VarSpeedServo. This will slow down its movement. The pitch, roll values from the android phone's orientation sensor are received by the Arduino to control the two servos which in turn tilts the labyrinth. The Bluetooth module forms the interface between the android and Arduino. The Arduino accepts the pitch, roll values from the Bluetooth and then constraints it to -/+ 15 degrees to keep the movement of labyrinth reasonable. Two Arduino sketch have been uploaded on Arduino IDE in order to control the movement of the servos and the labyrinth. In the sketch, first the VarSpeedServo objects for pitch and roll have been created then they are initialized to a value so as to make the direction levelled. For the labyrinth control sketch, the open serial communication via Bluetooth is enabled. In the loop, the Arduino checks for the availability of serial input. If found, it looks for the next valid integer of roll and pitch in the incoming serial stream and this goes on until it encounters the '\n' (new line) character. If '\n' is encountered, it comes out of the loop, disabling its connection with Bluetooth in the process.

The arduino sketch for controlling the labyrinth is as follows :

```
#include <VarSpeedServo.h>  //include the VarSpeedServo library
VarSpeedServo pitchServo;   //create VarSpeedServo objects for pitch and roll
VarSpeedServo rollServo;
const int svsp = 10;   //speed setting for VarSpeedServo SlowMove
const int pitchHm = 97;   //set so the maze pitch direction is level
const int rollHm = 90;    //set so the maze roll direction is level
int pitch;
int roll;

void setup()  //setup runs one time
{
  // Open serial communications-via bluetooth
  Serial.begin(9600);
```

```
pitchServo.attach(6);  //attach servos to pins
rollServo.attach(7);
pitchServo.slowmove(pitchHm, svsp);  //Home servos to start
rollServo.slowmove(rollHm, svsp);
delay(200);
 }
void loop() //loop repeats forever
{
  // if there's any serial available, read it:
 while (Serial.available() > 0) {
   // look for the next valid integer in the incoming serial stream:
   pitch = Serial.parseInt();
   // do it again:
   roll = Serial.parseInt();
   if (Serial.read() == '\n') break;  // look for the newline. That's the end of your read
 }
    //keep angles under 15 degrees
    pitch = constrain(pitch, -15, 15);
    roll = constrain(roll, -15, 15);
    //move servos to pitch and roll angles
    pitchServo.slowmove(pitchHm - pitch, svsp);  // change - to + if pitch is backword
    rollServo.slowmove(rollHm + roll, svsp);
    delay(70);  // let slowmove do its thing
}
```

The arduino sketch for controlling the servo motors is :

```
#include <VarSpeedServo.h>  //include the VarSpeedServo library
VarSpeedServo pitchServo;   //create VarSpeedServo objects for pitch and roll
VarSpeedServo rollServo;
const int svsp = 10;   //speed setting for VarSpeedServo SlowMove
const int pitchHm = 90;   //set so pitch direction is level
const int rollHm = 90;    //set so roll direction is level
```

```
void setup()  //setup runs one time
{
  pitchServo.attach(6);  //attach servos to pins
  rollServo.attach(7);

  pitchServo.slowmove(pitchHm, svsp);  //Home servos to start
  rollServo.slowmove(rollHm, svsp);
}

void loop() //loop repeats forever
{
  // Nothing here
}
```

3.3 BLUETOOTH TECHNOLOGY

HC-05 is a Bluetooth module. It utilizes Serial Port Protocol for wireless communication. It has an advantage of being used in both master and slave configurations. HC-05 is an important device for controlling the motors. Given its above mentioned advantages, it serves the purpose of an excellent solution for wireless communication in our project. The module has enhanced rate of Bluetooth V 2.0+ and also a modulation speed of 3 Mbps with a complete radio transceiver. Bluetooth consumes less power than other devices. HC-05 is employed for several applications like wireless communication between laptops, mobile phones, desktops and microcontrollers. It can also be used for consumer applications, data logging, robots and houses. HC-05 has the following default password- 1234 or 0000, default set to slave mode and data mode with a baud Rate of 9600. The HC-05 has a full-duplex communication and can be used to communicate with device having Bluetooth like laptop or phone or between microcontroller boards like Arduino. The module communicates using USART having a default baud rate of 9600. Therefore, interfacing with any microcontroller that supports USART is an easy task. We've used Arduino board as the microcontroller. Once the module is configured to command mode, it can transfer data from phone to microcontroller to phone. However, transferring of multimedia such as music, photos, videos is not possible using this module.

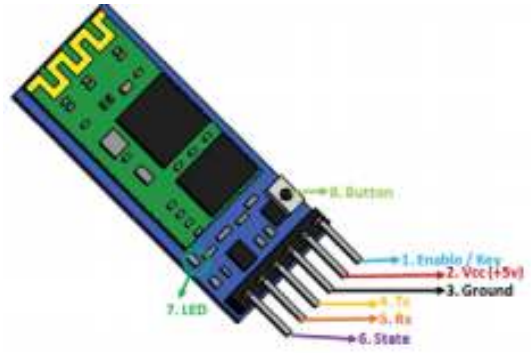


Fig 3.16 HC-05 Bluetooth Module

3.4 CIRCUIT CONNECTION

The two Servo Motors 1 and 2 are connected to the VCC and GND of the Arduino UNO each and to the pins 6 and 7 respectively.

The VCC and GND of the Bluetooth module is also connected to the VCC and GND of the Arduino UNO.

The Rx and Tx of Bluetooth Module is connected to the Tx and Rx of the Arduino UNO respectively.

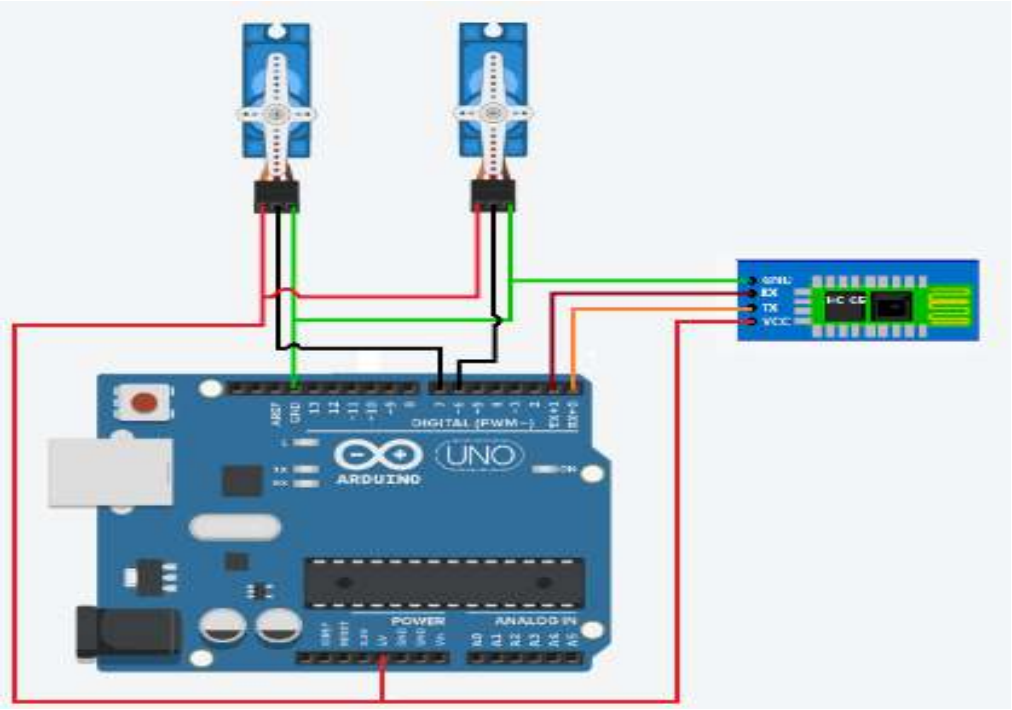


Fig 3.17 Circuit Connection

3.5 3D PRINTING

3D printing is sometimes referred to as Additive Manufacturing (AM). In 3D printing, one creates a design of an object using software, and the 3D printer creates the object by adding layer upon layer of material until the shape of the object is formed. It is a process in which arbitrarily configured objects are manufactured by depositing droplets of materials like polyamide (nylon), ABS- Acrylonitrile Butadiene Styrene plastic, PLA, glass filled stereo lithography materials (epoxy resins) or polyamide, stereo lithography materials (epoxy resins), silver, steel, photopolymers, titanium, wax and polycarbonate that gradually build up a 3-D object.

3D printers follow the instructions given by a computer to print a thing using the required materials like metal, ceramics or plastic. The whole procedure consists in creating an object layer by layer till it's ready. For example, one 3D printer may spray out melted half-liquid plastic that will get solid as each new layer appears.

The instructions are in the form of CAD files (computer-aided design) which are digital sketches used for creating different things. In fact any person can create a 3D model using the necessary software, connect the computer with the 3D printer and then wait for the machine to print the required object.

3.51 TYPES OF 3D PRINTING

1. Stereolithography (SLA)

Stereolithography makes use of a liquid plastic as the source material and this liquid plastic is transformed into a 3D object layer by layer[1]. Liquid resin is placed in a vat that has a transparent bottom. A UV (UltraViolet) laser traces a pattern on the liquid resin from the bottom of the vat to cure and solidify a layer of the resin. The solidified structure is progressively dragged up by a lifting platform while the laser forms a different pattern for each layer to create the desired shape of the 3D object.

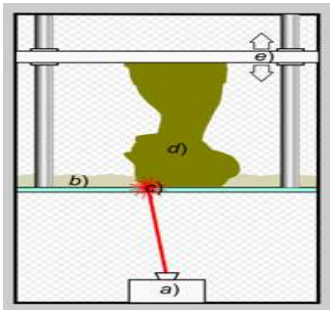


Fig 3.18Schematic representation of Stereolithography: a light-emitting device a) (a laser or DLP) selectively illuminates the transparent bottom c) of a tank b) filled with a liquid photo-polymerizing resin. The solidified resin d) is progressively dragged up by a lifting platform e) Source

2. Digital Light Processing (DLP)

3D printing DLP technology is very similar to Stereolithography but differs in that it uses a different light source and makes use of a liquid crystal display panel. This technology makes use of more conventional light sources and the light is controlled using micro mirrors to control the light incident on the surface of the object being printed. The liquid crystal display panel works as a photomask. This mechanism allows for a large amount of light to be projected onto the surface to be cured, thereby allowing the resin to harden quickly.

3. Fused Deposition Modeling (FDM)

With this technology, objects can be built with production-grade thermoplastics. Objects are built by heating a thermoplastic filament to its melting point and extruding the thermoplastic layer by layer.Special techniques can be used to create complex structures. For example, the printer can extrude a second material that will serve as support material for the object being formed during the printing process. This support material can later be removed or dissolved.

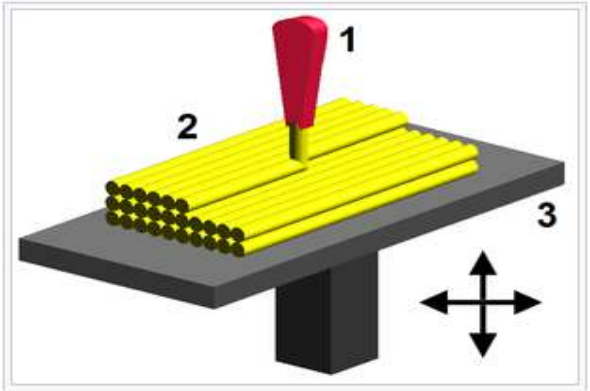


Fig 3.19Fused deposition modelling: 1) Nozzle ejecting molten material 2) Deposited material (modeledpart)3) Controlledmovable tableSource

4. Selective Laser Sintering (SLS)

SLS has some similarities with Stereolithography. However, SLS makes use of powdered material that is placed in a vat. For each layer, a layer of powdered material is placed on top of the previous layer using a roller and then the powdered material is laser sintered according to a certain pattern for building up the object to be created. Interestingly, the portion of the powdered material that is not sintered can be used to provide the support structure and this material can be removed after the object is formed for re-us

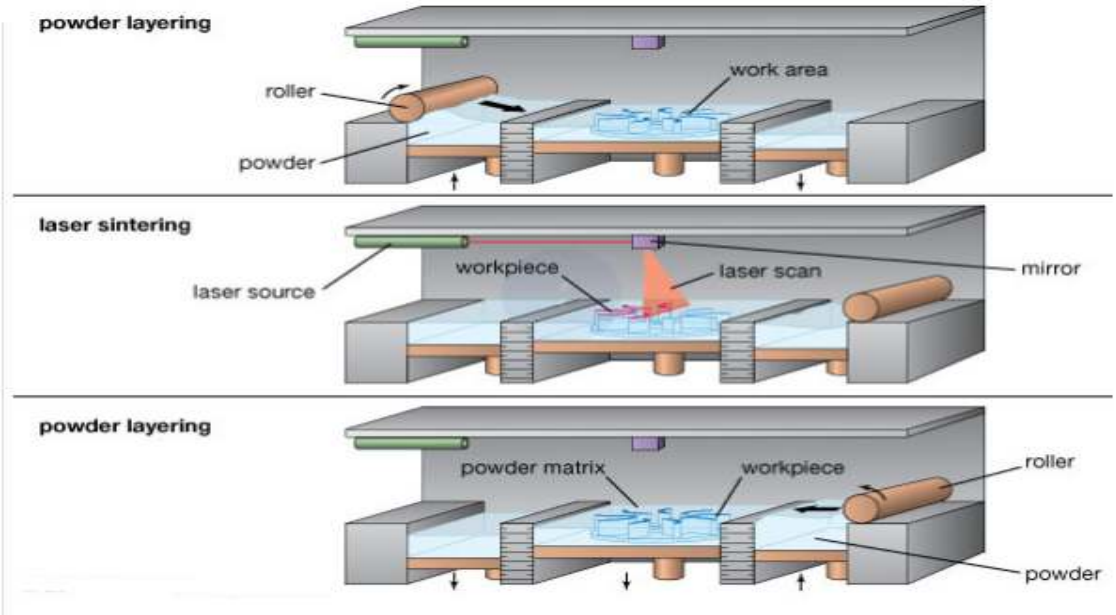


Fig 3.20Selective Laser Sintering Process

5. Selective Laser Melting (SLM)

The SLM process is very similar to the SLS process. However, unlike the SLS process where the powdered material is sintered the SLM process involves fully melting the powdered material.

6. Electronic Beam Melting (EBM)

This technology is also much like SLM. However, it makes use of an electron beam instead of a high-powered laser. The electron beam fully melts a metal powder to form the desired object. The process is slower and more expensive than for SLM with a greater limitation on the available materials.

7. Laminated Object Manufacturing (LOM)

This is a rapid prototyping system. In this process, layers of material coated with adhesive are fused together with heat and pressure and then cut into shape using a laser cutter or knife. More specifically, a foil coated with adhesive is overlaid on the previous layer and a heated roller heats the adhesive for adhesion between the two layers. Layers can be made of paper, plastic or metal laminates. The process can include post-processing steps that include machining and drilling.This is a fast and inexpensive method of 3D printing. With the use of an adhesion process, no chemical process is necessary and relatively large parts can be made.

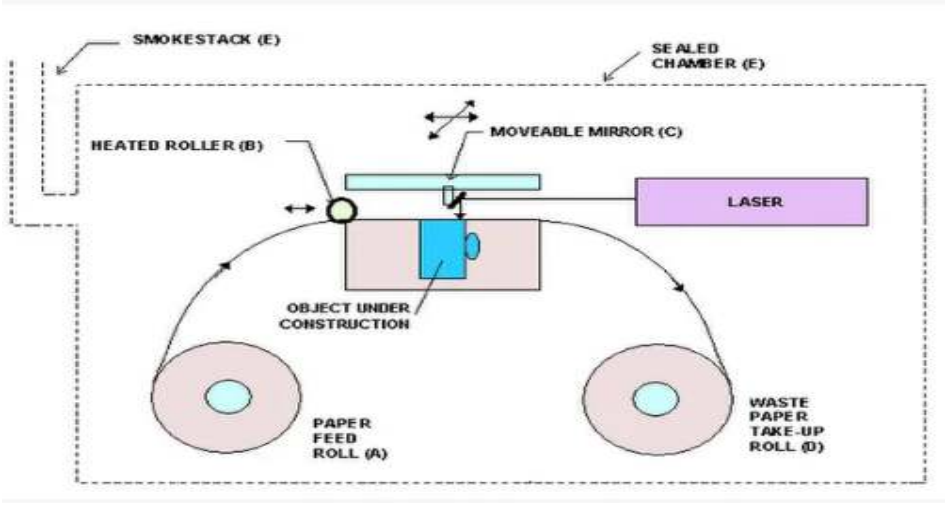


Fig 3.21Laminated Object Manufacturing

3.52 CONCEPT AND PRINCIPLE

To fully learn the concept and the principle of 3D printing, you should know these concepts or 3d printing terminology list and apply it on any 3D printable models or objects you want to print

1. Colors:

Choosing the right color for the model is also an important concept to consider. It makes the outer appearance of the 3D printed object, especially when the printed model is already assembled. For the precise color you want to achieve, check or search for it properly through different sites and books. For a more convenient color choosing or 3d printing multiple colors, be simple by choosing one from the basic or primary colors.

Fig 3.22 Colors

2. Tolerance:

This is an important concept to learn because it shows how each part fits together to create the piece. When creating parts that move, like the 3D printed ball joints, you will need to ensure that enough of the space is available for the tolerance.

Fig 3.23 Tolerance

3. Maximum Size:

Printing the 3D printed object, whether it is only a model or the actual piece, should be done in its maximum size. The object should be printed in the actual size to ensure that the output properly produced. In case the model you used is much bigger than the 3D printer you have, it is best to break down the printable model. The smaller pieces can be printed in a separate set and batch, or it could also be scaled to fit into the output.

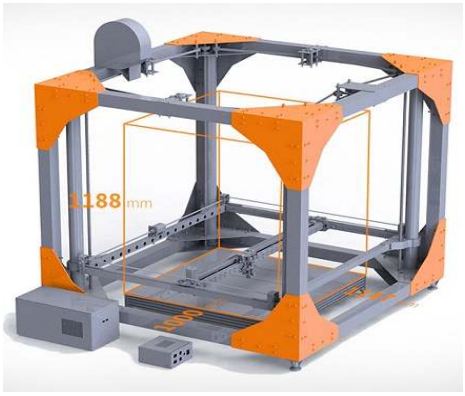


Fig 3.24 Maximum Size

4. Overhang:

Self-supported overhang is something that you can print without the need for the support, but it should be done accordingly to prevent problems. Too much over hangs is a big NO in printing because it may affect the object. There are times wherein you will not need to add overhangs and just leave the model as simple and easy to print as possible.

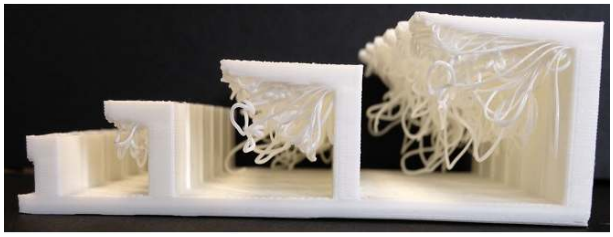


Fig 3.25 Overhang

5. Bridging:

Just as how the bridging is important, understanding its concept may give you an output that you want. The more spacious you allow the bridge to have, the more it will have pitfalls

when printing. One effective way to get through the bridges without causing it to fall is to print it upright.

Fig 3.26 Bridging

6. Supports:

A removable support is also an ideal way to provide a solution to bridging. 3D Printing with supports does not really mean you will have to do it in a more detailed way. If supports are needed for printed 3D model, 3D printing support material can be different from the main 3D printing filament. It is a good choice to use dissolvable support materials.

7. Polygons:

Compare the low and high resolution of the 3D printer. The polygons are important in that it keeps the design simple yet efficient. The low polygon models are actually more popular in the 3D confirmation, an the best about it is that it allows the designer to create objects that you can easily recognize.

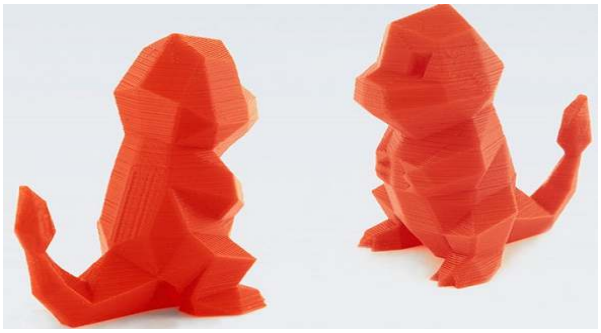


Fig 3.27 Polygons

8. XYZ:

Fig 3.28 XYZ

The XYZ representation is a familiar topic for math subjects. However, this is also an important part in any 3D modeling or in fabricating the object. As you work it with a CAD program, you will find it easier if you have the XYZ points.

9. Infill:

The infill also plays an important role when it comes to 3D printing and takes its place in 3D printing glossary. Although there is no need to keep the infill as hard and unbreakable as it is perceived, a simple honeycomb or diamond grid can be helpful to keep the object firm and steady.

The inner structure can be made denser or less, depending on what you required. This provides two benefits to the printing process. First is that it is convenient and easy to print. The second benefit is that it lessens the filament to be used for the object.

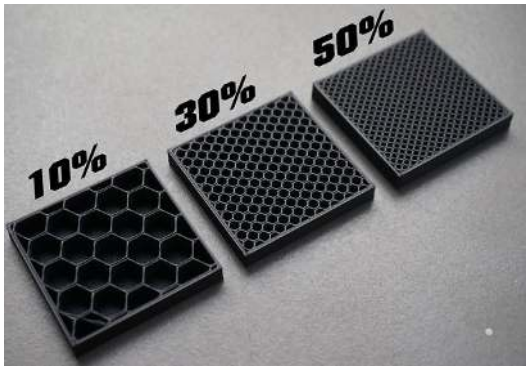


Fig 3.29 Infill

3.53 SOFTWARE TOOLS

There is many popular software tools or 3d printing software that can be used to create 3D models for 3D printer. You can find 3d printer software for the whole range of categories, such as 3d cad software, sculpting and freeform modeling software. Some of them are listed below.

Autodesk 123D Design:

Sketchup

3D Model To Print

Autodesk Maya

FreeCAD

Open SCAD

Sculptris

Photoshop CC

AutoCAD

Blender

Cinema 4D

And many more.
But the respective parts for our project is designed using the BLENDER.

3.54 BLENDER

Blender is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. Advanced users employ Blender's API for Python scripting to customize the application and write specialized tools; often these are included in Blender's future releases. Blender is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process. Examples from many Blender-based projects are available in the showcase.

Blender is cross-platform and runs equally well on Linux, Windows, and Macintosh computers. Its interface uses OpenGL to provide a consistent experience. To confirm specific compatibility, the list of supported platforms indicates those regularly tested by the development team.

As a community-driven project under the GNU General Public License (GPL), the public is empowered to make small and large changes to the code base, which leads to new features, responsive bug fixes, and better usability. Blender has no price tag, but you can invest, participate, and help to advance a powerful collaborative tool: Blender is your own 3D software. That is it is a Free Software, you are free to use Blender for any purpose, including commercially or for education.

What you create with Blender is your sole property. All your artwork – images or movie files – including the .blend files and other data files Blender can write, is free for you to use as you like. That means that Blender can be used commercially by artists, by studios to make animation films or VFX, by game artists to work on commercial games, by scientists for research, and by students in educational institutions.

Blender's GNU GPL license guarantees you this freedom. Nobody is ever permitted to take it away, in contrast to trial or "educational" versions of commercial software that will forbid your work in commercial situations.

3.55 THE FUTURE OF 3D PRINTING

3D printing is unlikely to replace the usual ways of making products. However this technology will offer the possibility of making individual parts whenever required – something more suitable for making uncommon parts for military aircraft, rather than making millions of tins for sale at a supermarket. Boeing has already made use of 3D printers for creating thousands of elements used on both military and civil craft flying nowadays.

3D printing is also being used in medical industry for making unique objects that are too difficult to make in a usual way. American surgeons have placed a 3D printed piece of skull that replaced 75 percent of the patient's skull. There has also been created a 3D printed ear template for a bioengineering ear with cells.The development of 3D printing all over the world would reduce its geography for home use and business. On-line markets make it possible for buyers to upload 3D printable templates of objects and sell them all over the world. Instead of paying shipping fees sellers can just order an object to be 3D printed at the nearest 3D printing shop for the customer. These 3D printing facilities will be available not only in special places.

It's not only businesses that will benefit from 3D-printing-on-demand. US militaries have

also made use of 3D printing labs in Afghanistan. It's a way of accelerating the pace of renewing and innovating battlefield and quickly supplying soldiers with what they need. NASA look to 3D printing to make it possible to build replacement parts or spaceships in orbit.Many of the 3D printers don't exceed the sizes of usual home appliances like fridges, however some of them can be as big as your house.

3.56 RESTRICTIONS OF 3D PRINTING

Nevertheless 3D printing is not without its restrictions. The majority of the printers can create an object only of a specific material. That is what prevents 3D printers from making complex things like Apple products. Luckily some companies have started thinking of workarounds. A New Mexico company named Optomec has made a 3D printer able to print a microcircuit chip into an object.

3D printing boost may turn both positive and negative. The ability of easy sharing sketches online and printing things at home, for example, is an advantage for DIY enthusiasts.

Experts on security are worried that 3D printing provides the ways to increase digital piracy and to share knowledge that would be fatal if used in the wrong way. The Texas group Defense Distributed has extended the social boundaries developing the first ever 3D printed gun.

To set up the labyrinth framework, there is a demand to pick a pertinent material that is rigid, light weight, friction resistant and easy to manufacture. Therefore, the labyrinth board parts can be designed and manufactured independently by employing the 3D printing process.

3.57 3D PRINTING PROCESS

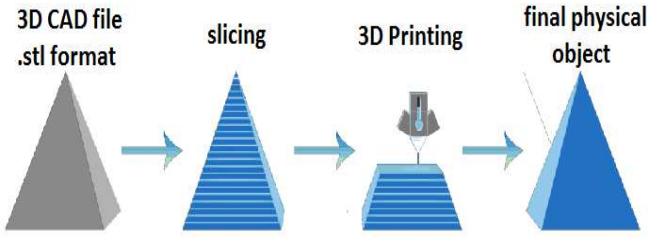


Fig 3.30 3D Printing Process

First we need to design the different parts , and this is done using the Blender software.

When starting Blender, the splash screen appears in the center of the window. It contains options to create new projects or open recently opened blend-files. To close the splash screen and start a new project, click anywhere outside the splash screen (but inside the Blender Window) or press Esc. The splash screen will disappear revealing the default screen.

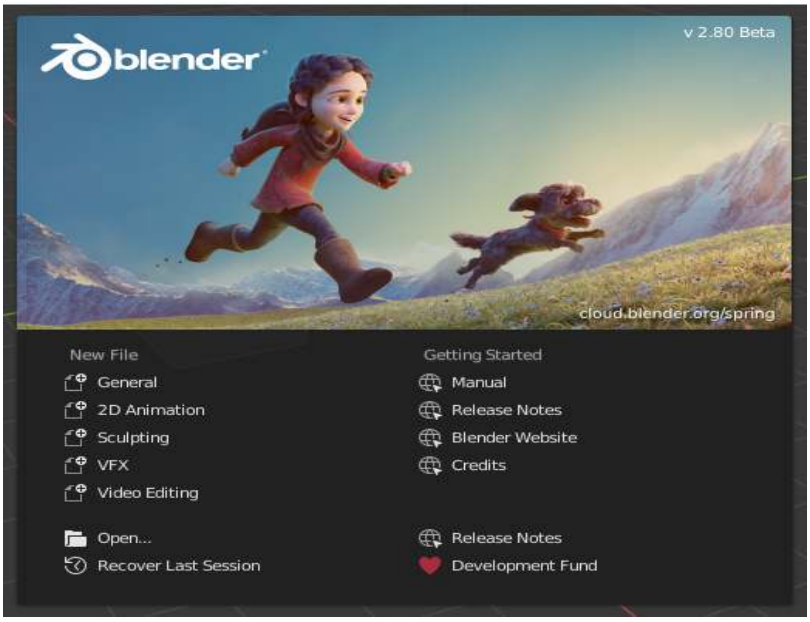


Fig 3.31 Blender

After starting Blender and closing the Splash Screen the Blender window should look something similar to the image below; as Blender's user interface is consistent across all platforms.

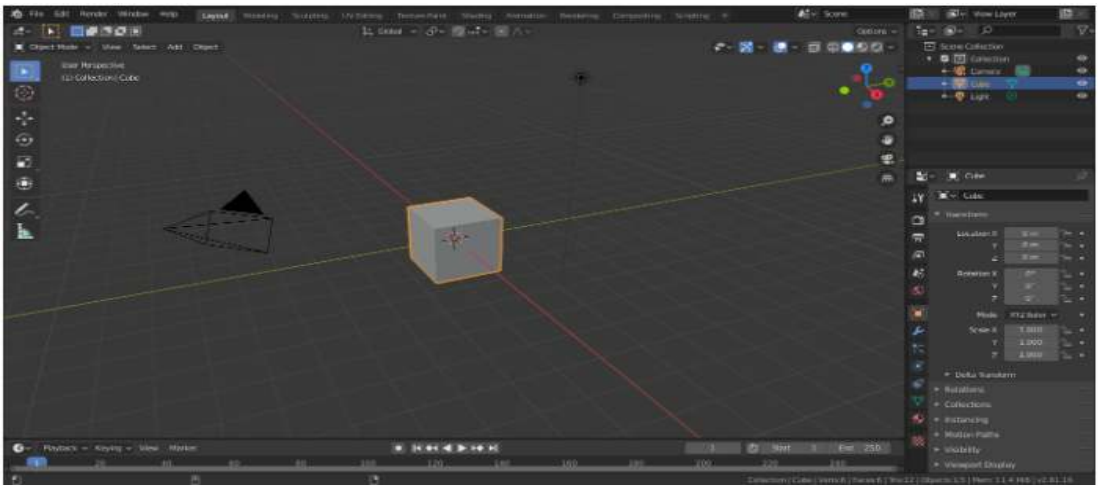


Fig 3.32 Blender's Interface

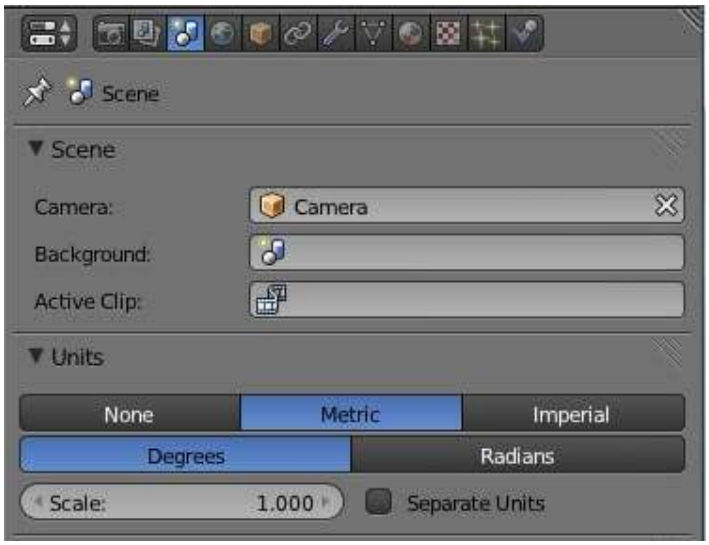
Blender's interface is separated into three main parts:

- Topbar at the very top.
- Areas in the middle.
- Status Bar at the bottom.

Preparing Blender Software

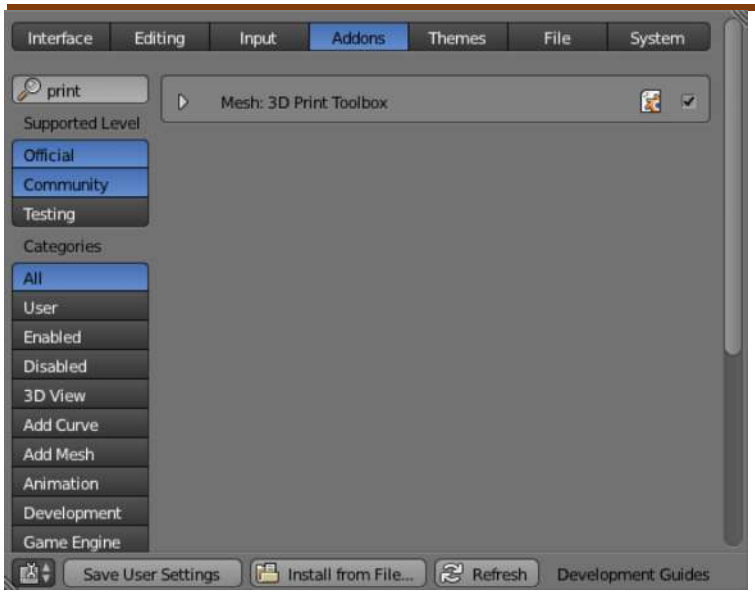
1. Adjusting Scale and Units of Measurement

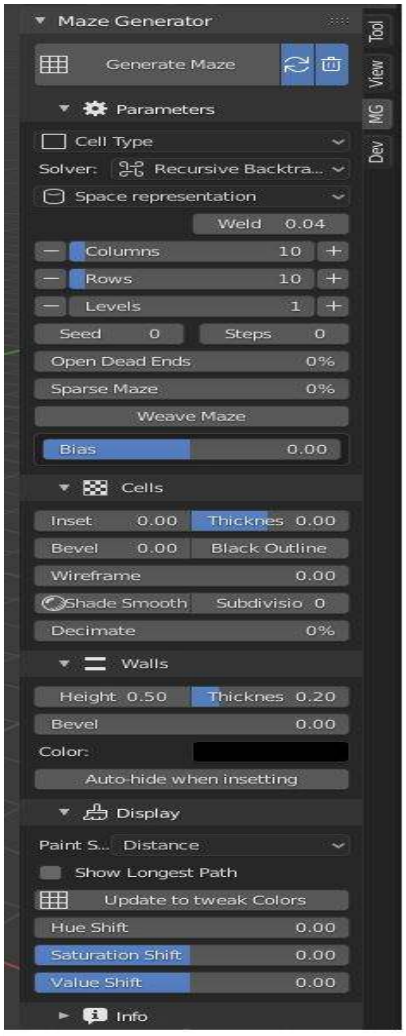
By default, Blender uses an inherent system of measurement which is not based on a measurable unit. However, as the object you will be designing will be physically manufactured, it's important to define a unit of scale. To accomplish this, head to the "Properties" bar and select "Scene" which is represented by a small sphere, cylindar and light. There you will find measurement options under "Units" — choose "Metric". By default, one "Blender" unit is equivalent to 1 meter. The "Scale" parameter allows you to raise or lower that equivalence globally. You'll find the size of the object in the object's properties (found by typing 'N').



Activating Addons

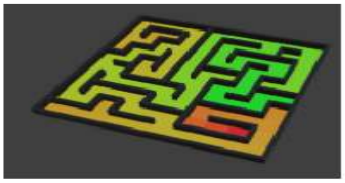
Certain Addons will be used and it is important to turn them on. To do this, go into User Preferences (Ctrl+Alt+U) and into the "Addons" section. From there use the search function to find the necessary Addon. Addons to search for and Activate: "Looptools" and "3D PrintToolbox" To keep these Addons activated by default, click on "Save User Settings".





Cell types : Square

Space representations : Plane



Algorithms :

We will generate a maze in Blender Python using the following steps…

1. Create a large plane, partitioned in NxN squares at Z=0
2. Set our starting point to the bottom left corner of the plane
3. Move in a random direction, forward, backward, left, or right
4. If the tile we are "standing" on is at Z=0, depress it (or extrude it down) by H units
5. Repeat steps 3 and 4 until we have cut a path through the plane and are "standing" B units away from the maze

With the right parameters, this should leave us with a cool maze. It will help to add another parameter that specifies the probability of a forward step versus a backward step, as we will see. Flipping some parameters around, we should be able to adjust it to make catwalks and multi-level structures.

2. Selection by Location

This is a feature I believe should be native to Blender. It involves selecting vertices, edges, or faces of a Blender objects according to their location in 3D space. The version we will discuss here is for selecting faces. Writing this algorithm requires a good understanding of Blender's internal data structures.

3. Extrusion

Extrusion is Blender's word for pulling a vertex, edge, or face outward while keeping it attached to the object. Extrusion can normally be accessed as an Edit Mode function, but we will access through Blender Python with the bpy.ops.mesh.extrude_region_move() function.

4. Subdivision

Blender has a handy function for splitting up planes with an arbitrary number of sides into more similarly-shaped planes. This does not change the shape of the plane, but allows us to create a grid out of plane consisting of a single face. We will access this through the bpy.ops.mesh.subdivide() function.

.

3.58 CREATING THE LABYRINTH

Using the above functions and outline, we can create mazes and catwalks with the following

```
import bpy
import random
# _____________________ MAZE GENERATOR _______________
# Choose your settings, then click "Run Script" in the bottom of
# this window to make a custom maze!
#
# NOTE: The entrance is the hole in the 3rd quadrant (-x, -y), and
# the exit is in the first quadrant (+x, +y). These mazes are more
# difficult to solve if you start at the correct entrance. In the
# future, I may somehow mark the entrance and exits, but for now,
# I don't really know any good ways to do so! Leave a comment on
# Thingiverse if you have an idea!

#~~~~~~~~~~~~~~~~~~~~~USER SETTINGS ~~~~~~~~~~~~~~~~~~~~~~~~~

width = 20              #how many cells wide
length = 20             #how many cells long
cellThickness = 4         #width of passages [mm]
wallThickness = 1         #width of walls [mm]
wallHeight = 6            #height of walls [mm] (0 = flat path)
baseHeight = 1            #height of base [mm] (0 = no base)
punchEntranceExit = True    #if entrance and exit will be open
dualExtrusion = False       #if base will be seperate from maze

# ~~~~~~~~~~~~~~~~~~END USER SETTINGS ~~~~~~~~~~~~~~~~~~~~~~
#       ANYTHING BELOW THIS LINE IS ONLY PROGRAMMER NOTES
# ____________________________________________________________
#first, delete everything in the scene in order to start clean
```

```
bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete()

#initialize arrays and such
directions = ["n", "s", "e", "w"]
stack = []
currentCell = [width - 1, length - 1] #start at the end of the maze, makes exit dead end
explored = [[False for y in range(length)] for x in range(width)] #make the explored array [X][Y]

#make the array of vertices
verts = [] #initial
#c = 0 (bottom left) squares grid
for y in range(0, length + 1):
    for x in range(0, width + 1):
        verts.append( ((wallThickness + cellThickness) * x, (wallThickness + cellThickness) * y, 0) )
#c = 1 (bottom right) squares grid
for y in range(0, length + 1):
    for x in range(0, width + 1):
        verts.append( (((wallThickness + cellThickness) * x) + wallThickness, (wallThickness + cellThickness) * y, 0) )
#c = 2 (top left) squares grid
for y in range(0, length + 1):
    for x in range(0, width + 1):
        verts.append( ((wallThickness + cellThickness) * x, ((wallThickness + cellThickness) * y) + wallThickness, 0) )
#c = 3 (top right) squares grid
for y in range(0, length + 1):
    for x in range(0, width + 1):
        verts.append( (((wallThickness + cellThickness) * x) + wallThickness, ((wallThickness + cellThickness) * y) + wallThickness, 0) )

#let's give ourselves an easy way to get the index of a vert, because it's hella complicated
def vertIndex(x, y, c):
```

```
    return (c * ((length + 1) * (width + 1))) + (((width + 1) * y) + x)

#now, make the initial grid faces
faces = [] #initial
#first, the horizontals
for y in range(0, length + 1):
    for x in range(0, width):
        faces.append( (vertIndex(x, y, 1), vertIndex(x + 1, y, 0), vertIndex(x + 1, y, 2), vertIndex(x, y, 3)) )
#then, the verticals
for y in range(0, length):
    for x in range(0, width + 1):
        faces.append( (vertIndex(x, y, 2), vertIndex(x, y, 3), vertIndex(x, y + 1, 1), vertIndex(x, y + 1, 0)) )
#finally, the squares themselves
for y in range(0, length + 1):
    for x in range(0, width + 1):
        faces.append( (vertIndex(x, y, 0), vertIndex(x, y, 1), vertIndex(x, y, 3), vertIndex(x, y, 2)) )

#make the array that will be used to signal a face to be deleted
faceToDelete = [False for n in range(0, len(faces))]

#function to test if it's legal to move in a given direction
def canMove(dir):
        if dir == "n":
            if currentCell[1] < length - 1:
                if explored[currentCell[0]][currentCell[1] + 1]:
                    return False
                else:
                    return True
            else:
                return False

        elif dir == "s":
```

```
        if currentCell[1] > 0:
            if explored[currentCell[0]][currentCell[1] - 1]:
                return False
            else:
                return True
        else:
            return False

    elif dir == "e":
        if currentCell[0] < width - 1:
            if explored[currentCell[0] + 1][currentCell[1]]:
                return False
            else:
                return True
        else:
            return False

    elif dir == "w":
        if currentCell[0] > 0:
            if explored[currentCell[0] - 1][currentCell[1]]:
                return False
            else:
                return True
        else:
            return False

#function to move and mark the wall in the way for death
def move(dir):
  if dir == "n":
    faceToDelete[(width * (currentCell[1] + 1)) + currentCell[0]] = True
    currentCell[1] += 1
  elif dir == "s":
    faceToDelete[(width * currentCell[1]) + currentCell[0]] = True
    currentCell[1] -= 1
  elif dir == "e":
```

```
        faceToDelete[(width * (length + 1)) + ((width + 1) * currentCell[1]) + currentCell[0] +
1] = True
        currentCell[0] += 1
    elif dir == "w":
        faceToDelete[(width * (length + 1)) + ((width + 1) * currentCell[1]) + currentCell[0]]
= True
        currentCell[0] -= 1

#main recursive carving loop

#would have used do..while loop, but Python doesn't have that :(
keepRunning = True
while keepRunning:
    #make explored
    explored[currentCell[0]][currentCell[1]] = True

    #randomize directions
    random.shuffle(directions)

    #movement switch
    if canMove(directions[0]):
        stack.append(currentCell[:])
        move(directions[0])
    elif canMove(directions[1]):
        stack.append(currentCell[:])
        move(directions[1])
    elif canMove(directions[2]):
        stack.append(currentCell[:])
        move(directions[2])
    elif canMove(directions[3]):
        stack.append(currentCell[:])
        move(directions[3])
    else: #if no direction can be moved into
        currentCell = stack.pop()
```

```
    #finally, check if stack is empty
    keepRunning = (len(stack) != 0)

if punchEntranceExit:
    #mark the entrance and exit for death
    faceToDelete[0] = True #entrance
    faceToDelete[((length + 1) * width) - 1] = True #exit

#last vert/face operation, delete the faces marked for death
for n in range(len(faces), 0, -1):
    if faceToDelete[n - 1]:
        faces.pop(n - 1)

# _____ FINAL MESH CREATION BLOCK _____
mazeMesh = bpy.data.meshes.new("MazeMesh") #create a new mesh

mazeObj = bpy.data.objects.new("Maze", mazeMesh) #create an object with that mesh
mazeObj.location = (0, 0, 0) #position object at the origin
bpy.context.scene.objects.link(mazeObj) #link object to scene

#fill the mesh with verts, edges, faces
mazeMesh.from_pydata(verts,[],faces)     #edges or faces should be [], or you ask for problems
mazeMesh.update(calc_edges=True)    #update mesh with new data

# ######## editing mode block #1 - extrusion ########
bpy.context.scene.objects.active = bpy.data.objects["Maze"] #set as active
if wallHeight > 0:
    bpy.ops.object.editmode_toggle()

    #extrude to wallHeight
    bpy.ops.mesh.select_all(action='SELECT') #select all
    bpy.ops.mesh.extrude_region_move(MESH_OT_extrude_region={"mirror":False},
TRANSFORM_OT_translate={"value":(0, 0, wallHeight), "constraint_axis":(False, False,
True), "constraint_orientation":'NORMAL', "mirror":False, "proportional":'DISABLED',
```

```
"proportional_edit_falloff":'SMOOTH', "proportional_size":1, "snap":False, "snap_target":'CLOSEST', "snap_point":(0, 0, 0), "snap_align":False, "snap_normal":(0, 0, 0), "texture_space":False, "remove_on_cancel":False, "release_confirm":False})

    #fix normals
    bpy.ops.mesh.select_all(action='SELECT') #select all
    bpy.ops.mesh.normals_make_consistent(inside=False)
    bpy.ops.object.editmode_toggle()
    # ######## object mode ########
    bpy.ops.object.select_all(action='SELECT') #select all

    #add floor, if needed
    if baseHeight > 0:
        bpy.context.object.location[2] = baseHeight #move up
        #add the base cube
        bpy.ops.mesh.primitive_cube_add(radius=.5, location=(((width * cellThickness) + ((width + 1) * wallThickness)) / 2, ((length * cellThickness) + ((length + 1) * wallThickness)) / 2, baseHeight / 2))
        bpy.ops.transform.resize(value=(((width * cellThickness) + ((width + 1) * wallThickness)), ((length * cellThickness) + ((length + 1) * wallThickness)), baseHeight))
        for obj in bpy.context.selected_objects:
            obj.name = "Base"
        bpy.context.scene.objects.active = bpy.data.objects["Maze"] #set as active
        if not dualExtrusion:
            bpy.ops.object.modifier_add(type='BOOLEAN')
            bpy.context.object.modifiers["Boolean"].operation = 'UNION'
            bpy.context.object.modifiers["Boolean"].object = bpy.data.objects["Base"]
            bpy.ops.object.modifier_apply(apply_as='DATA', modifier="Boolean")
            bpy.ops.object.select_pattern(pattern="Base")
            bpy.ops.object.delete(use_global=False)

# ######## editing mode block #2 - simplify final mesh ########
bpy.ops.object.editmode_toggle()
bpy.ops.mesh.select_all(action='SELECT')
bpy.ops.mesh.normals_make_consistent(inside=False)
```

```
bpy.ops.mesh.dissolve_limited()
bpy.ops.object.editmode_toggle()
######### object mode ########
#center it for 3D printing
bpy.ops.object.select_all(action='SELECT')
bpy.context.scene.cursor_location = [((width * cellThickness) + ((width + 1) * wallThickness)) / 2, ((length * cellThickness) + ((length + 1) * wallThickness)) / 2, 0]
bpy.ops.object.origin_set(type='ORIGIN_CURSOR')
bpy.context.object.location = [0, 0, 0]
if (dualExtrusion and (wallHeight > 0)):
    bpy.context.scene.objects.active = bpy.data.objects["Base"] #set as active
    bpy.ops.object.origin_set(type='ORIGIN_CURSOR')
    bpy.context.object.location = [0, 0, 0]
bpy.context.scene.cursor_location = [0, 0, 0]
if wallHeight <= 0:
    bpy.ops.object.convert(target='CURVE')
    bpy.ops.object.select_all(action='DESELECT')
```

To make maze, download `Maze Generator.blend`, open it in Blender, and read the instructions in the script in the upper left.

When it's done, select it, then *File>Export>Stl (.stl)*. Print in 3d printer.

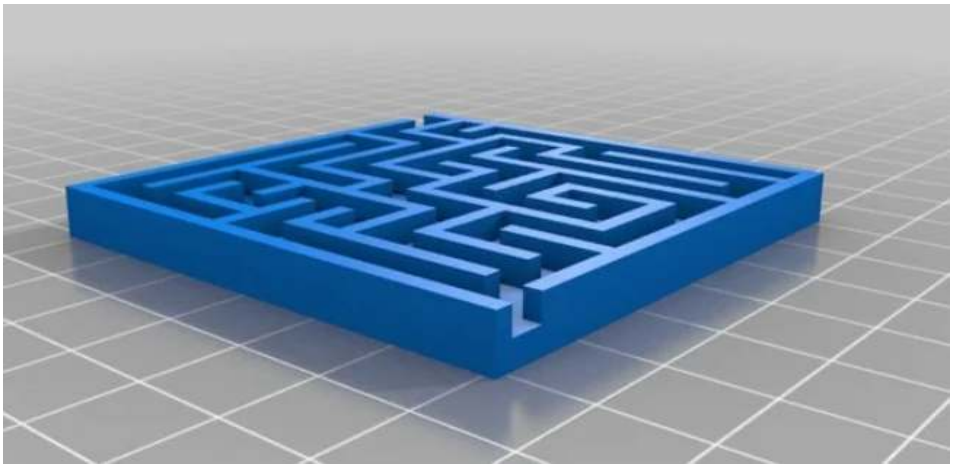


Fig 3.33 stl maze file

The wall width is made multiple of 3d printer nozzle diameter so you can print them solid using only shells. As the nozzle width was 0.4 mm, we printed the walls solidly using 1 shell *(which will make the walls 2 shells thick total)* if they are 0.4*2 mm thick. Alternatively, we can laser cut them! Support for this is limited ATM, but if set to the `wallHeight = 0`, it will make a path *(really, 2 paths)* instead of a mesh. We use the

Export/Inkscape_SVG_Exporter) to get it as an SVG and of course, it's not fit to laser cut right out of the box.*(If you use Inkscape, scale the paths by 125% before using them. Illustrator works as-is.)* .Currently, the entrance and exit look the same. The entrance is the hole that will face towards you when printed *(or the one in the 3rd quadrant [-x, -y], for those who want to be specific)*.

Similarly other parts of the set up is also designed.

The most significant advantage of this approach is that the designer's imagination need not have boundaries.

3D Printed Parts for the Frame Work:

As shown in the figure, to set up the labyrinth model the following parts are built

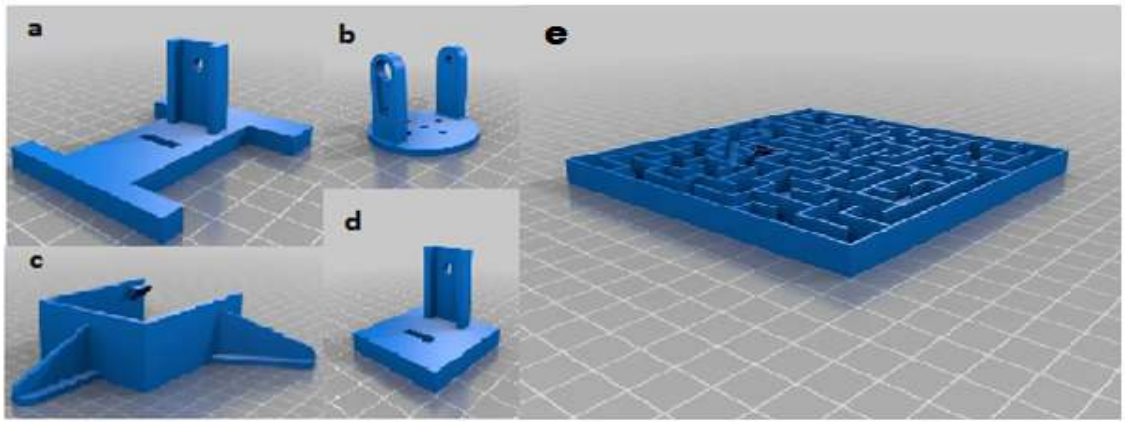


Fig 3.34 The stl files of parts to set up the labyrinth game a) Gimbal base- on which the roll servo motors are placed. b) Gimbal center- to which the rotor is attached. c) Labyrinth base. d) Gimbal top- holds the pitch servo motor. e) Labyrinth

There are 2 servo motors used to make the whole set up working. Each of the motors are attached to gimbals as shown in the figure. Where the servos are attached to the gimbal base and the gimbal top is attached to the servo rotors with the help of the screws for easy and smooth motion.

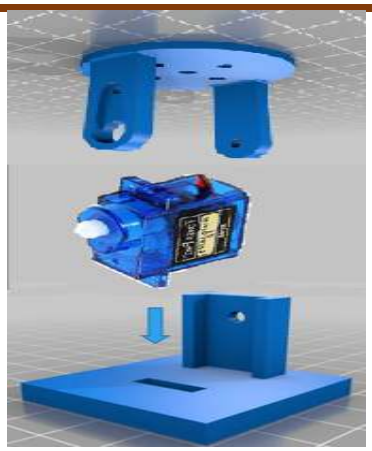


Fig 3.35 Servo Motor Set Up With The Gimbal Parts.

Of the two servo motors, one is the pitch servo and the other is the roll servo motor. The pitch servo is responsible for the left and right direction motion and the roll servo is responsible for the front and back directions.

The above set up for both the motors is established and are placed one above the other where the rotors of each motors are directed orthogonal to each other so as to sweep all the four directions.

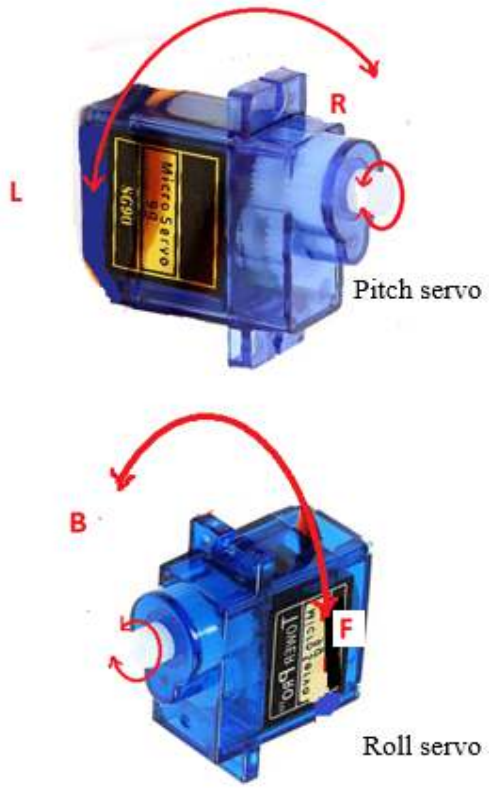


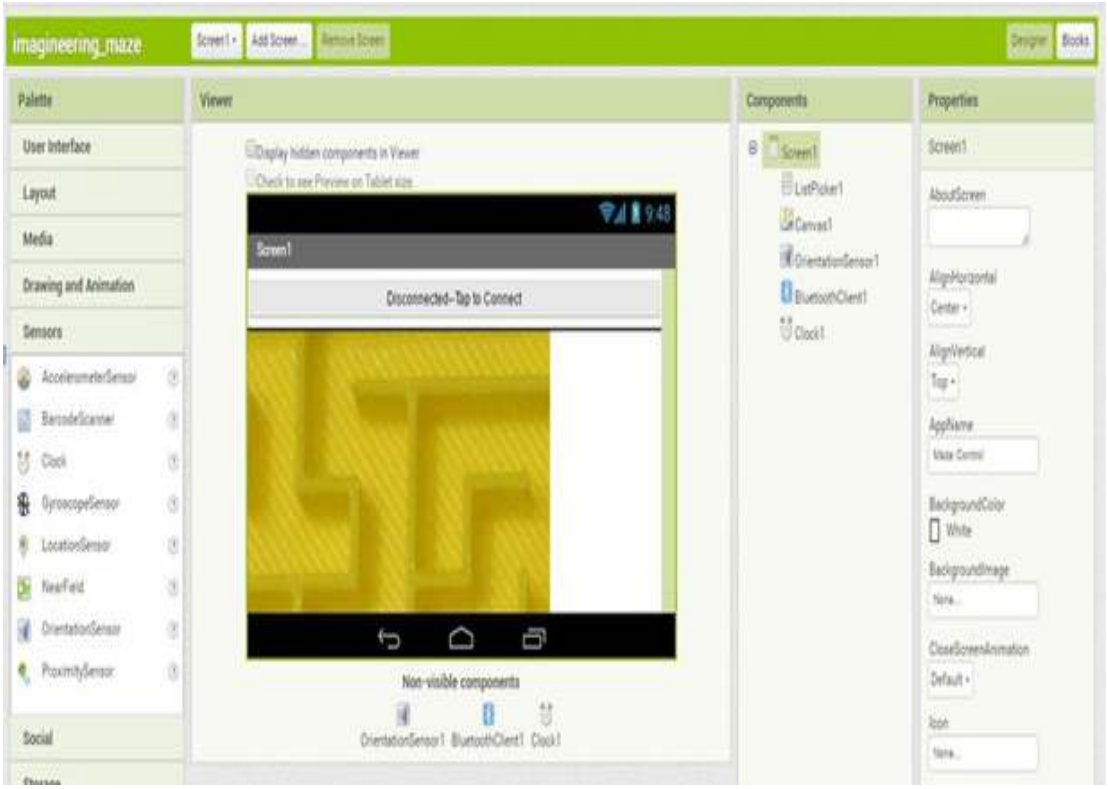
Fig 3.36Servo Motor Assembly

And above this the labyrinth board is fixed which will move according to the servo motor rotation. And hence the set up is ready.

Chapter 4

RESULTS AND DISCUSSIONS

We have developed the logic controller of the labyrinth using the MIT App Inventor 2 and also designed the 3D printing blocks to form the labyrinth board. The Blocks code on MIT App Inventor is as follows:

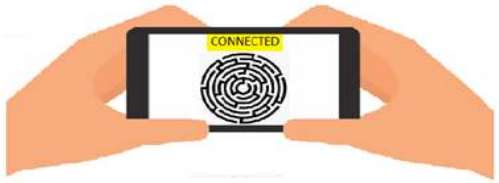


The following table gives the cost incurred for each components used in the project:

1.	3D Printed Labyrinth and other parts	Rs.900
2.	Arduino Uno	Rs.345
3.	HC-05 Bluetooth module	Rs.230
4.	Cable	Rs.60
5.	Rivets, metal ball, screws and nuts	Rs.100
6.	SG-90 Servo motors	Rs.190

TOTAL COST INCURRED : Rs. 1825/-

Below figure shows the final model of the project developed.



Chapter 5

APPLICATIONS

The game provides the following benefits to the player:

- Improving hand-eye coordination
- Solving mazes also boosts patience and persistence.
- Improves spatial logical skill.
- Provides an exciting gaming experience.
- Provides skill developing knowledge to children by implementing it in schools as an application of GDF (Game Development Framework).

Chapter 6

CONCLUSION

In today's world, reliance on wireless control is very much necessary for everyone. In our project, we have used Bluetooth technology as the wireless technology so that it can control the movement of the labyrinth and have made use of Android phone (a commodity being easily available). For this, the desired application can be installed in the player's phone. Then the player has to switch ON the Bluetooth in their mobile. The player can rotate the android phone in various directions in order to tilt and control the labyrinth board. The Bluetooth sends these commands to the Arduino so that the two motors can be controlled. Using 3D Printing, any complex design of the maze can be created. It gives a better quality of view and is more sustainable. The culmination of an android device and game board as in our project allows the player to focus more on the playfield rather than the device screen. This game can help in providing skill developing knowledge to children by implementing it in schools as an application of GDF (Game Development Framework).

REFERENCES

[1]https://english.stackexchange.com/questions/144052/difference-between-labyrinth-and-maze

[2]https://www.makerbot.com/stories/engineering/advantages-of-3d-printing/

[3]https://www.peertechz.com/articles/doi10.17352-2455-3484.000025-jamts.php

[4] Android-based wireless controller for military robot using Bluetooth Technology- By Widodo Buiharto, Jarot Sembodo Suroso.

[5] MIT App Inventor Cheat Sheet

[6] appinventor.pevest.com

[7]https://docs.blender.org/manual/en/latest/interface/splash.html

[8] https://docs.blender.org/manual/en/latest/index.html

[9] Ahmad AdamuGaladima "Arduino as a learning tool", 2014 IEEE

[10] Subankar Roy , TashiRapdenWangchuk, Rajesh Bhatt "Arduino Based Bluetooth Controlled Robot", International Journal of Engineering Trends and Technology (IJETT) – Volume 32 Number 5- February 2016