Processor Architecture Lab and Open source technology Lab

TITLE:
Motion Gaming Controller

Name	**Roll number**
Dipesh Bharambe	06
Mihir Chitre	10
Shubham Sapkal	56

INTRODUCTION :

This arduino project uses the integration of the python language and the concepts on the arduino board to create a game in Python and a controller for it. We have used the **'pygame'** module in python to create a game with the help of the GUI functions available in the module.

The game is a simple **Breakout** game, in which a ball keeps moving on the screen and the aim is to break all the bricks on the screen. We also need to ensure that the ball does not touch the lower edge of the screen. If this happens then the game ends and the player loses.

The ball can be prevented from touching the lower edge by moving the paddle provided on the screen. This paddle is controlled by hand gestures. When the ball touches the paddle, it bounces back in the opposite direction.

The controller is to be worn on the hand. When the hand is tilted on the left side, the paddle moves towards the left. When the hand is tilted on the right side, the paddle moves towards right. The hand can be swiped over the sensor provided to start or pause the game.

OBJECTIVE :

Nowadays, gaming is becoming a very highly pursued industry. A lot of people have been introduced to gaming and are adopting it to reduce stress and for entertainment. However too much gaming results in lack of exercise and various illnesses are become very prominent.

One of the most significant problem, which is faced world-wide is obesity. People refrain from outdoor exercises and prefer gaming instead. Thus our project aims at making gaming a healthier choice for the gamers. Motion gaming will not only help gamers to exercise but it will also make a lot of games more interactive and fun.

Motion gaming can highly improve the body and eye co-ordination of the individuals. Motion controllers can also be useful in various simulators. Thus the main objective of our project is to make the available softwares more interactive and to give a real-time experience to the users.

Hardware Used :

This project uses the following hardware components :

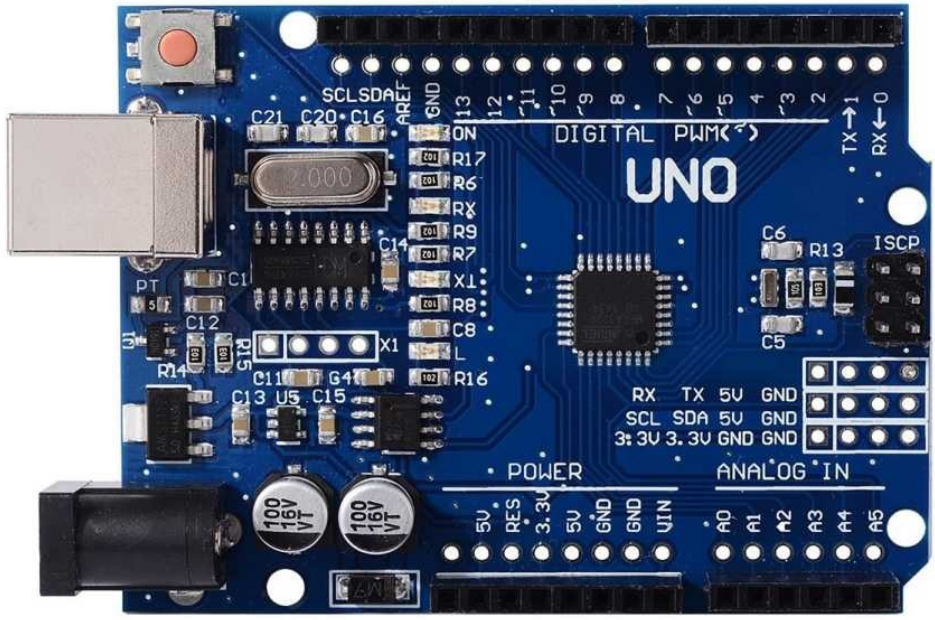
- Arduino uno
- HC-304 Ultrasonic sensor.
- ADXL335 Accelerometer
- Bread board
- Male-Male connecting wires.
- Male-Female connecting wires.

1. Arduino Uno :

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection,

a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform.



2. Ultrasonic Sensor :

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object using the formula :

$$distance = \frac{speed\ of\ sound\ \times time\ taken}{2}$$ Speed of sound = 340 m / s

The trig pin in the sensor generates the sound waves and the echo pin in the sensor catches the reflected sound waves.

This project uses the ultrasonic sensor to find the distance of the hand from the sensor. When the hand is moved closer to the sensor, the sensor sends a signal due to which the game is paused if the game is being played or a new game is started if the has been lost or won.

3. Accelerometer :

An accelerometer measures proper acceleration, which is the acceleration it experiences relative to freefall and is the acceleration felt by people and objects. Put another way, at any point in spacetime the equivalence principle guarantees the existence of a local inertial frame, and an accelerometer measures the acceleration relative to that frame. Such accelerations are popularly denoted g-force; i.e., in comparison to standard gravity.

An accelerometer at rest relative to the Earth's surface will indicate approximately 1 g *upwards*, because any point on the Earth's surface is accelerating upwards relative to the local inertial frame (the frame of a freely falling object near the surface). To obtain the acceleration due to motion with respect to the Earth, this "gravity offset" must be subtracted and corrections made for effects caused by the Earth's rotation relative to the inertial frame.

We have used the accelerometer to find the change in the acceleration of the hand. This acceleration obtained is first converted into a unit g-force which the unit of earth’s gravitaional pull, using the formula :

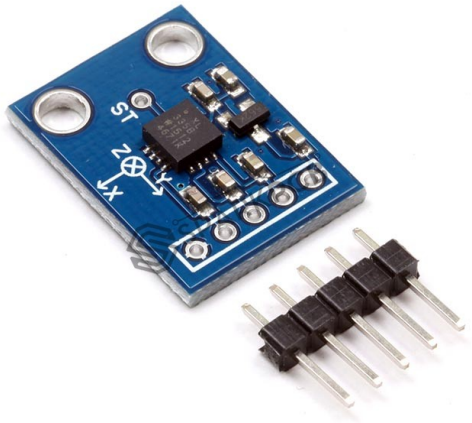
$$\mathbf{Aout} = \frac{\frac{\mathbf{ADC\ value\ *\ Vref}}{\mathbf{1024}} \mathbf{-\ Voltage\ Level\ at\ 0g}}{\mathbf{Sensitivity\ Scale\ Factor}}$$

Then these values of acceleration along X,Y and Z axes is used to find the angle of rotation along X-axis also known as **‘Roll’**.

*roll = (((atan2(y_g_value,z_g_value) * 180) / 3.14) + 180)*

When the hand is tilted towards left, the value of **roll** below 50, and when the hand is tilted towards right the value of the roll goes above 300. These changes in the value of roll are used to move the paddle in the game.

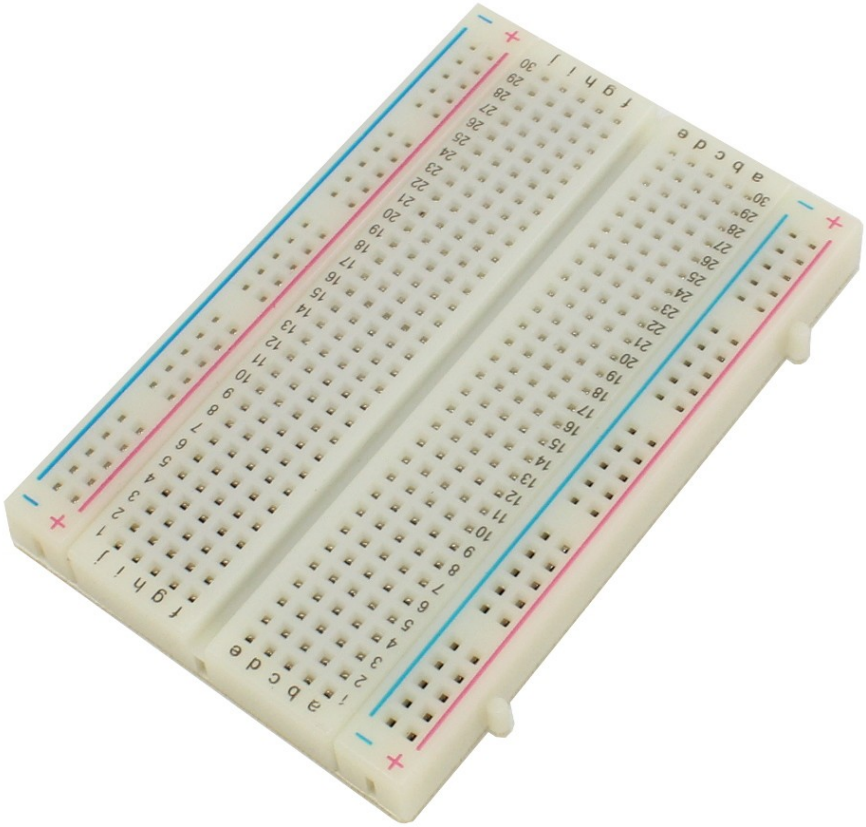
When the value of roll is between 50-300 there is no movement of the paddle. This indicates that there is no motion of the hand.



4. Bread Board :

A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically.

We have used the breadboard to connect the accelerometer and ultrasonic sensor to the **Vcc** and the **GND** of the arduino uno and to provide a stable base for accelerometer when mounted on the hand.



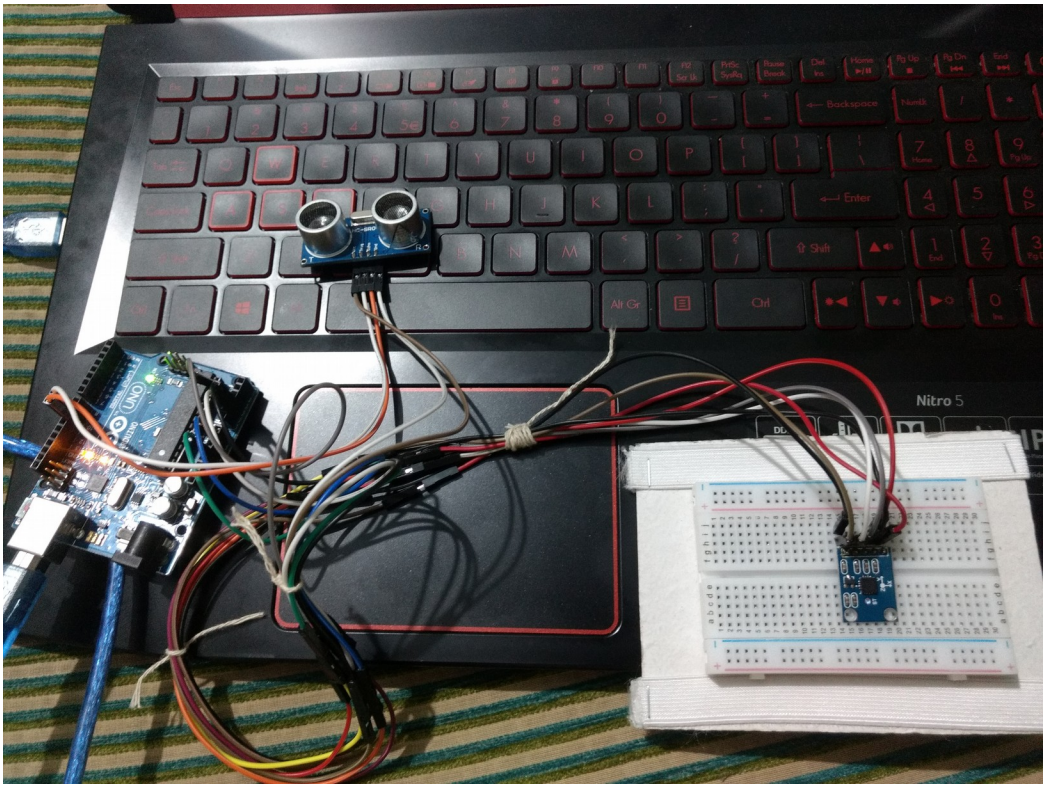
5. **Male-Male connecting wires :**

These wires are used to make connections from the bread board to the arduino board.

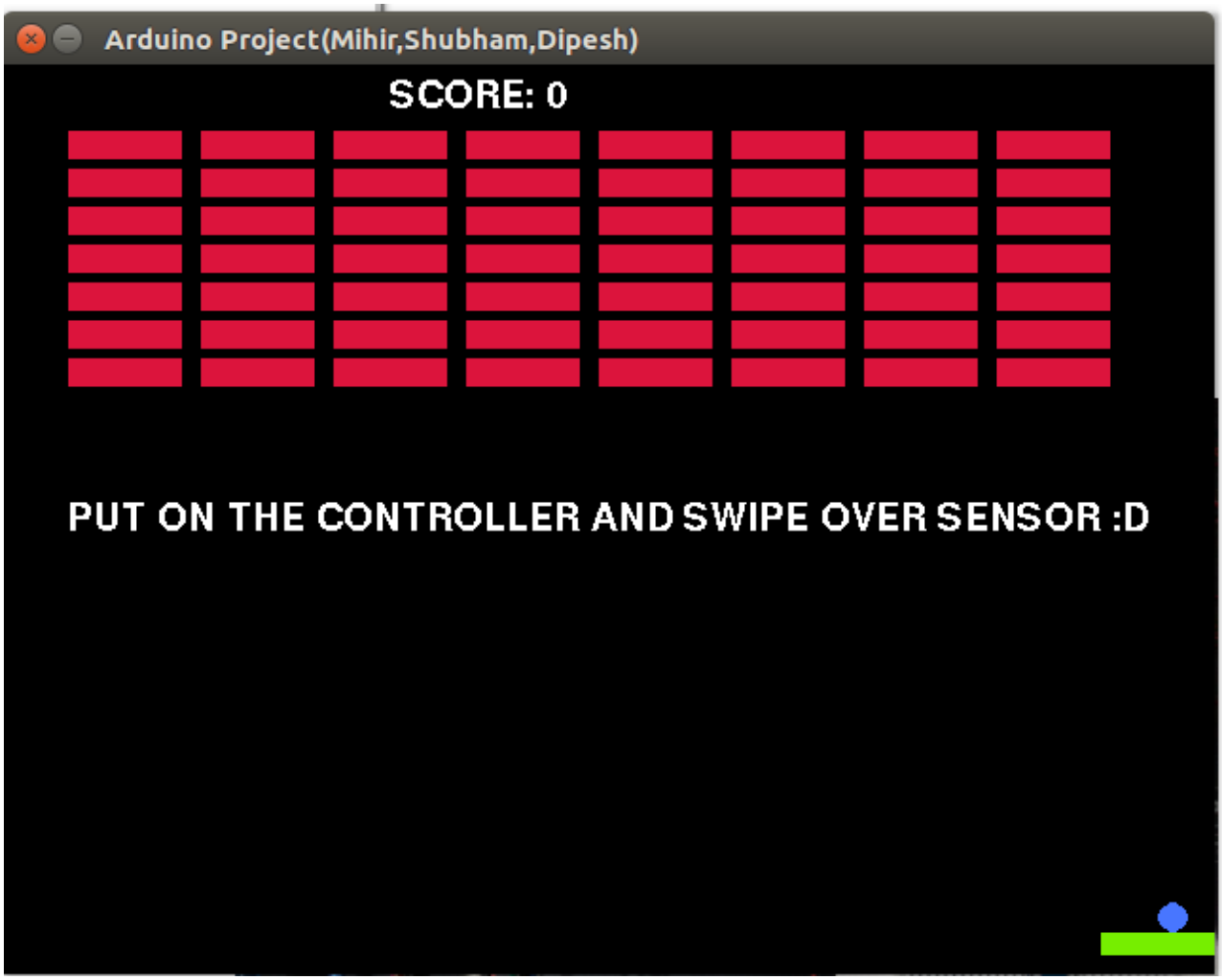
6. **Male-Female connecting wires :**

These wires are used as an extension to increase the distance between the arduino board and the controller to provide ease of movement.

ACTUAL SETUP :



GAME WINDOW :



PROJECT CODES :

1. **Arduino code :**

```
#include<math.h>

const int x_out = A1;

const int y_out = A2;

const int z_out = A3;

const int trigPin = 9;

const int echoPin = 10;

long duration;

int distance;


void setup() {

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
```

```
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600);
}

void loop() {
  int x_adc_value, y_adc_value, z_adc_value;
  double x_g_value, y_g_value, z_g_value;
  double roll;
  x_adc_value = analogRead(x_out);
  y_adc_value = analogRead(y_out);
  z_adc_value = analogRead(z_out);

  x_g_value = ( ( ( (double)(x_adc_value * 5)/1024) - 1.65 ) / 0.330 ); /* Acceleration in x-direction in g units
*/
  y_g_value = ( ( ( (double)(y_adc_value * 5)/1024) - 1.65 ) / 0.330 ); /* Acceleration in y-direction in g units
*/
  z_g_value = ( ( ( (double)(z_adc_value * 5)/1024) - 1.80 ) / 0.330 ); /* Acceleration in z-direction in g units
*/

  roll = ( ( (atan2(y_g_value,z_g_value) * 180) / 3.14 ) + 180 ); /* Formula for roll */

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
// Calculating the distance
  distance= duration*0.034/2;
  if(distance<=10)
  {   distance=(-1)*distance;
      Serial.println(distance);
```

```
  }
  else
  {   Serial.println(roll);
  }
  delay(50);
}
```

Python code :

```
#Breakout using arduino
import sys
import pygame
import serial

arduino=serial.Serial('/dev/ttyACM0',9600)

SCREEN_SIZE   = [640,480]

BRICK_WIDTH   = 60
BRICK_HEIGHT  = 15
PADDLE_WIDTH  = 60
PADDLE_HEIGHT = 12
BALL_DIAMETER = 16
BALL_RADIUS   = BALL_DIAMETER / 2

MAX_PADDLE_X = SCREEN_SIZE[0] - PADDLE_WIDTH
MAX_BALL_X   = SCREEN_SIZE[0] - BALL_DIAMETER
MAX_BALL_Y   = SCREEN_SIZE[1] - BALL_DIAMETER

PADDLE_Y = SCREEN_SIZE[1] - PADDLE_HEIGHT - 10

BLACK = (0,0,0)
WHITE = (255,255,255)
BALL = (72,118,255)
PADDLE  = (118,238,0)
```

```
BRICK_COLOR = (220,20,60)

STATE_BALL_IN_PADDLE = 0
STATE_PLAYING = 1
STATE_WON = 2
STATE_GAME_OVER = 3
STATE_PAUSE = 4

class breakout:

    def __init__(self):
        pygame.init()

        self.screen = pygame.display.set_mode(SCREEN_SIZE)
        pygame.display.set_caption("Arduino Project(Mihir,Shubham,Dipesh)")

        self.clock = pygame.time.Clock()

        if pygame.font:
            self.font = pygame.font.Font(None,30)
        else:
            self.font = None

        self.init_game()



    def init_game(self):
        self.score = 0
        self.state = STATE_BALL_IN_PADDLE

        self.paddle   = pygame.Rect(300,PADDLE_Y,PADDLE_WIDTH,PADDLE_HEIGHT)
        self.ball     = pygame.Rect(300,PADDLE_Y -
BALL_DIAMETER,BALL_DIAMETER,BALL_DIAMETER)
```

```
    self.ball_vel = [5,-5]

    self.create_bricks()


def create_bricks(self):
    y_ofs = 35
    self.bricks = []
    for i in range(7):
        x_ofs = 35
        for j in range(8):
            self.bricks.append(pygame.Rect(x_ofs,y_ofs,BRICK_WIDTH,BRICK_HEIGHT))
            x_ofs += BRICK_WIDTH + 10
        y_ofs += BRICK_HEIGHT + 5

def draw_bricks(self):
    for brick in self.bricks:
        pygame.draw.rect(self.screen, BRICK_COLOR, brick)

def check_input(self):
                intn=str(arduino.readline())
                intn=float(intn)
                if 0<intn<=150.00:
                        self.paddle.left -= 5
                        if self.paddle.left < 0:
                                self.paddle.left = 0

                if intn>=200.00:
                        self.paddle.left += 5
                        if self.paddle.left > MAX_PADDLE_X:
                                self.paddle.left = MAX_PADDLE_X
```

```
                if intn<0.00 and self.state == STATE_PLAYING:
                        self.state=STATE_PAUSE

                if intn>=0.00 and self.state == STATE_PAUSE:
                        self.state=STATE_PLAYING

                if intn<0.00 and self.state == STATE_BALL_IN_PADDLE:
                        self.ball_vel = [5,-5]
                        self.state = STATE_PLAYING

                if intn<0.00 and (self.state == STATE_GAME_OVER or self.state == STATE_WON):
                        self.init_game()

def move_ball(self):
    self.ball.left += self.ball_vel[0]
    self.ball.top  += self.ball_vel[1]

    if self.ball.left <= 0:
        self.ball.left = 0
        self.ball_vel[0] = -self.ball_vel[0]
    elif self.ball.left >= MAX_BALL_X:
        self.ball.left = MAX_BALL_X
        self.ball_vel[0] = -self.ball_vel[0]

    if self.ball.top < 0:
        self.ball.top = 0
        self.ball_vel[1] = -self.ball_vel[1]
    elif self.ball.top >= MAX_BALL_Y:
        self.ball.top = MAX_BALL_Y
        self.ball_vel[1] = -self.ball_vel[1]

def handle_collisions(self):
    for brick in self.bricks:
```

```
            if self.ball.colliderect(brick):
                self.score += 10
                self.ball_vel[1] = -self.ball_vel[1]
                self.bricks.remove(brick)
                break

        if len(self.bricks) == 0:
            self.state = STATE_WON

        if self.ball.colliderect(self.paddle):
            self.ball.top = PADDLE_Y - BALL_DIAMETER
            self.ball_vel[1] = -self.ball_vel[1]
        elif self.ball.top > self.paddle.top:
            self.state = STATE_GAME_OVER

    def show_stats(self):
        if self.font:
            font_surface = self.font.render("SCORE: " + str(self.score), False, WHITE)
            self.screen.blit(font_surface, (205,5))

    def show_message(self,message):
        if self.font:
            size = self.font.size(message)
            font_surface = self.font.render(message,False, WHITE)
            x = (SCREEN_SIZE[0] - size[0]) / 2
            y = (SCREEN_SIZE[1] - size[1]) / 2
            self.screen.blit(font_surface, (x,y))


    def run(self):
        while 1:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
```

```
                sys.exit


            self.clock.tick(50)
            self.screen.fill(BLACK)
            self.check_input()

            if self.state == STATE_PLAYING:
                self.move_ball()
                self.handle_collisions()
            elif self.state == STATE_BALL_IN_PADDLE:
                self.ball.left = self.paddle.left + self.paddle.width / 2
                self.ball.top  = self.paddle.top - self.ball.height
                self.show_message("PUT ON THE CONTROLLER AND SWIPE OVER SENSOR :D")
            elif self.state == STATE_GAME_OVER:
                self.show_message("..GAME OVER..SWIPE TO PLAY AGAIN")
            elif self.state == STATE_WON:
                self.show_message("..YOU WON..SWIPE TO PLAY AGAIN")
            elif self.state == STATE_PAUSE:
                self.show_message("..PAUSED..REMOVE THE HAND OVER SENSOR TO CONTINUE")

            self.draw_bricks()

            pygame.draw.rect(self.screen, PADDLE , self.paddle)

            pygame.draw.circle(self.screen, BALL, (self.ball.left + BALL_RADIUS, self.ball.top +
BALL_RADIUS), BALL_RADIUS)

            self.show_stats()

            pygame.display.flip()

if __name__ == "__main__":
    breakout().run()
```

PYTHON MODULES USED :

1. **SYS Module :**

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

This project uses the **sys** module to handle all the system calls made by the python game.

These system calls include the creation of an application window, the keyboard simulations which change due to the change in values of the sensor and to end the process of the game when a command to close the game window is given.

2. **PYGAME Module :**

This module helps in creating GUI based applications in Python. This module has inbuilt functions to draw various geometrical figures and objects on the application window which makes it very easy to create games.

This project uses the **pygame** module to create the bricks, the paddle and the ball after creating an application window of the specified size.

3. **PYSERIAL Module :**

This module helps the python program to communicate through serial port. In this project the values of the ultrasonic sensor and the accelerometer obtained via arduino uno are sent to the python program through the serial port.

FUTURE SCOPE :

Currently, this project uses only the hand movements to control the game. This project has the capacity to change the way people perceive gaming by programming the movements of entire body to control the virtual objects.

Sensors can be attached to legs and head to simulate the process of running by making the player run actually in real life. This will increase the people’s interest in gaming as well as result in more entertainment.

This will also help athletes to test their limits and their abilities even in unavailability of grounds or open spaces. We also aim at integrating this movement based system with the concept of virtual reality to capture the entire world in a virtual progam. Virtual Reality combined with motion control might give a way to a lot of new inventions.

Thus, this project is made with an attempt to make the people more active.

PROBLEMS FACED :

The major problem faced while creating this interactive gaming experience is integrating game together with the gaming controller.

At an early stage, this project aimed at creating a controller for already established and famous games which are played by people all over the world. But as this project progessed , it was realised that most of the games do not take into account the python programs running in background. Hence as a result, it is difficult to make the changes is the game according to the change in values of the sensor.

Thus, this project also had to include a development of the game which will support python and change according to the values obtained in the python program.

Another problem faced is that python and the arduino board are set to communicate with each other through the serial port. As there is only one serial port in arduino uno, only one sensor can send its value through the serial port at a time. Also, when the value reaches the python code, it is difficult for the python code to interpret which values is from the ultrasonic sensor and which value is from the accelerometer.

This problem was overcomed by handling all the calculations of ultrasonic sensor in terms of negative values.

REFERENCES :

- http://www.electronicwings.com/arduino/adxl335-accelerometer-interfacing-with-arduino-uno
- https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/
- https://howtomechatronics.com/projects/arduino-game-controller/
- https://pythonhosted.org/pyserial/
- http://pyautogui.readthedocs.io/en/latest/
- https://www.pygame.org/docs/