Smartie homie

By Goldwires

Vu Nguyen (Alex), Chen Shih (Oscar), Younsuk Choi, Phan Thanh Long, Usame Gonul (Sam)

Aim of Goldwires

- Improve interactivity of the smart home systems and the user.
- Improve the user’s quality of life and living standard.
- Multiple subsystems available for customization.

Hardware consideration

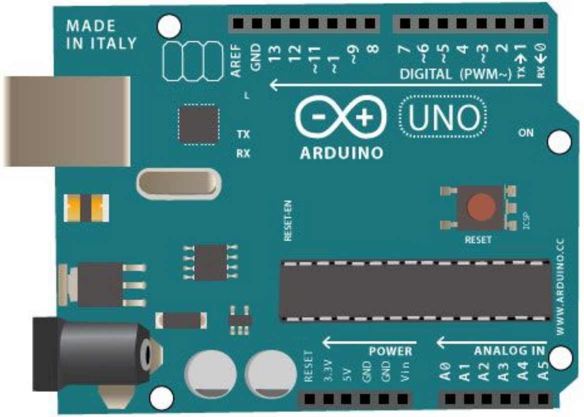


Fig. 1 Arduino UNO [1]

- Taking into account the type of systems that being designed, the choice for hardware parts can vary drastically.
- Arduino Uno was ultimately chosen since it is the most accessible and the only microcontroller that is available on Tinkercad.
- Simulate the microcontroller as cost saving measure while team members can collaborate remotely.

Device manager

- The center of the smart home system.
- Receive and relay commands to the other parts of the systems.
- Help user with inquiries, act as a speaker, or control devices remotely.

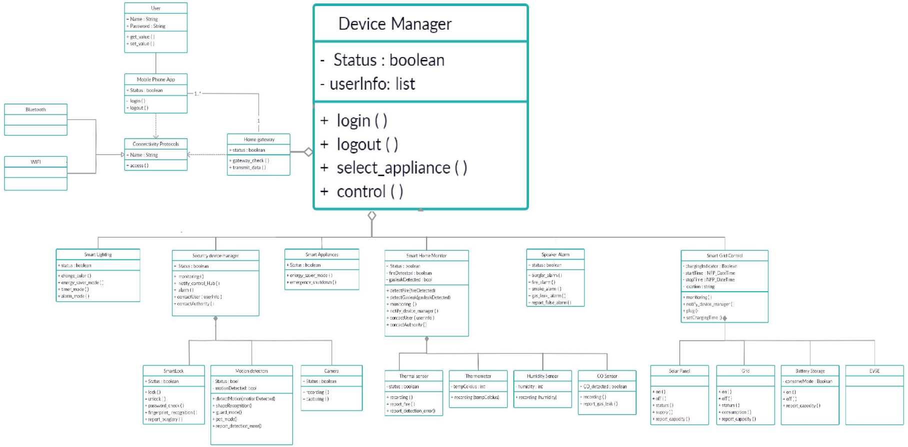
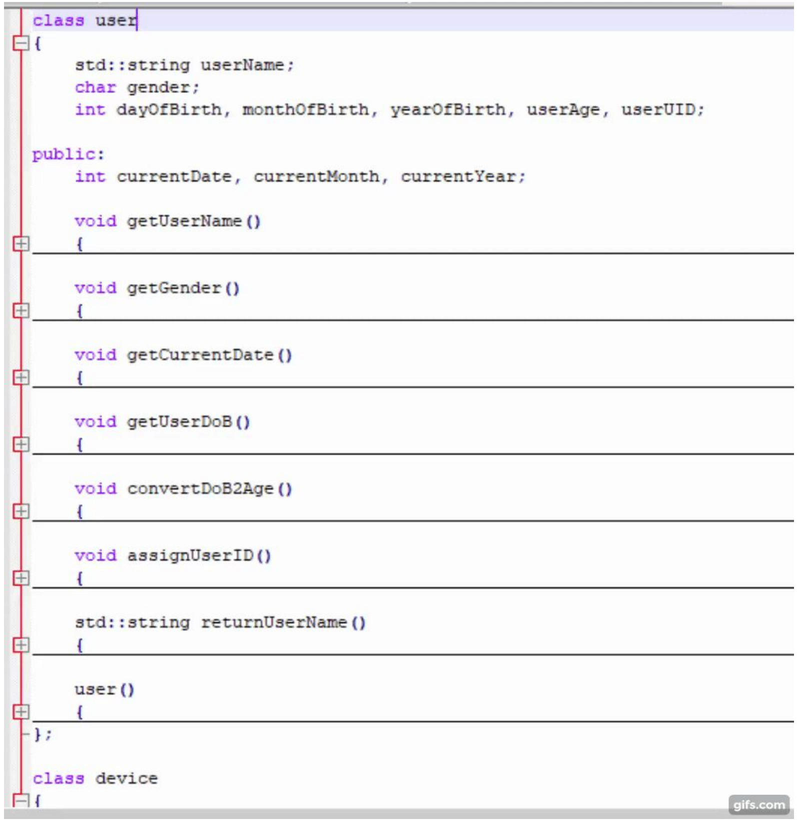


Fig. 2.1.1 Class diagram



```
class user
{
    std::string userName;
    char gender;
    int dayOfBirth, monthOfBirth, yearOfBirth, userAge, userUID;

public:
    int currentDate, currentMonth, currentYear;

    void getUserName()
    {

    void getGender()
    {

    void getCurrentDate()
    {

    void getUserDoB()
    {

    void convertDoB2Age()
    {

    void assignUserID()
    {

    std::string returnUserName()
    {

    user()
    {
};

class device
{
```

Fig. 2.1.2 Realisation using Classes and Vectors

Device manager - State machine

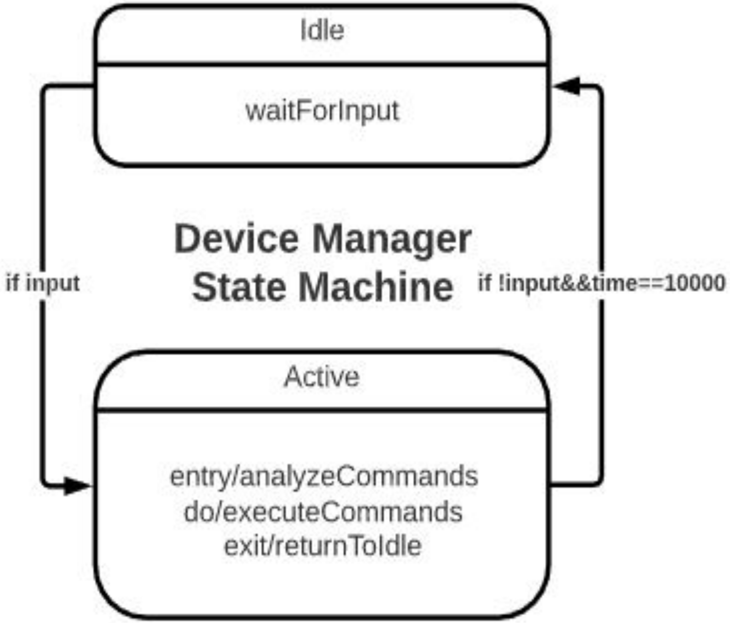
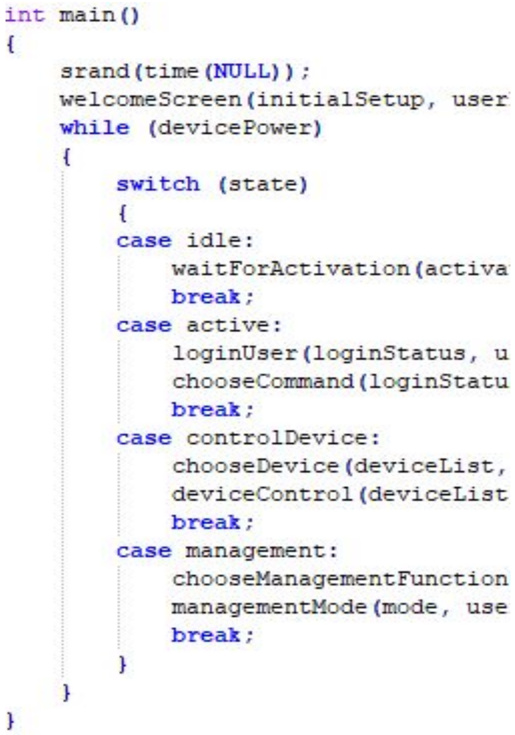
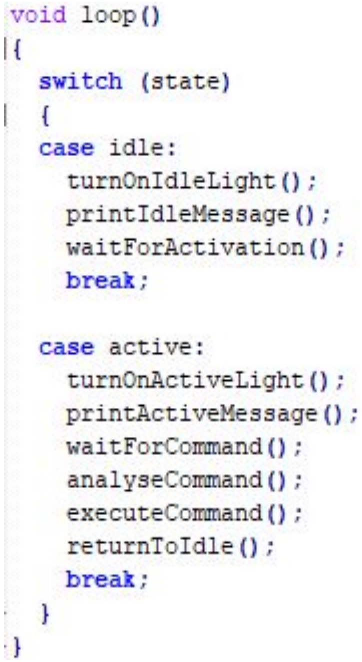


Fig. 2.2.1 State machine diagram



```
int main()
{
    srand(time(NULL));
    welcomeScreen(initialSetup, user
    while (devicePower)
    {
        switch (state)
        {
        case idle:
            waitForActivation(activa
            break;
        case active:
            loginUser(loginStatus, u
            chooseCommand(loginStatu
            break;
        case controlDevice:
            chooseDevice(deviceList,
            deviceControl(deviceList
            break;
        case management:
            chooseManagementFunction
            managementMode(mode, use
            break;
        }
    }
}
```

Fig. 2.2.2 PC implementation code



```
void loop()
{
  switch (state)
  {
  case idle:
    turnOnIdleLight();
    printIdleMessage();
    waitForActivation();
    break;

  case active:
    turnOnActiveLight();
    printActiveMessage();
    waitForCommand();
    analyseCommand();
    executeCommand();
    returnToIdle();
    break;
  }
}
```

Fig. 2.2.3 Arduino code

Device Manager- Simulation

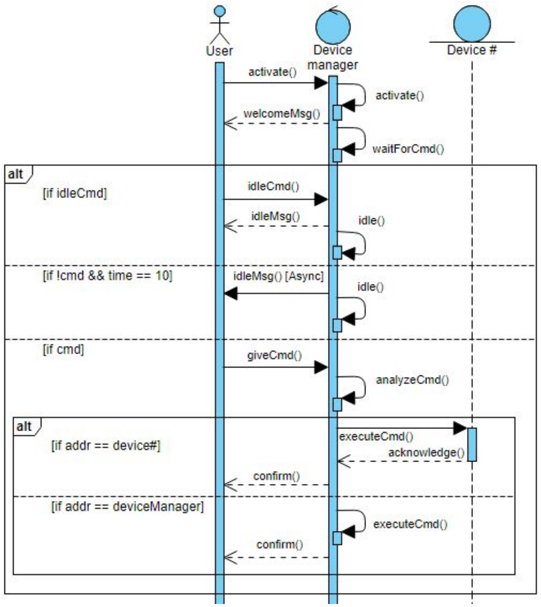


Fig. 2.3.1 User sequence diagram

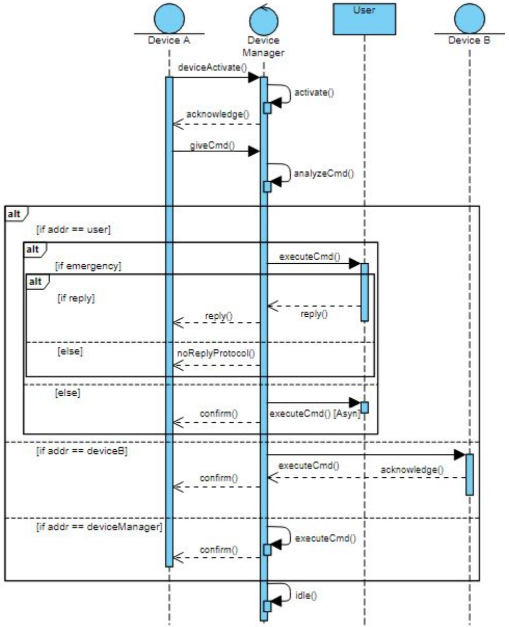
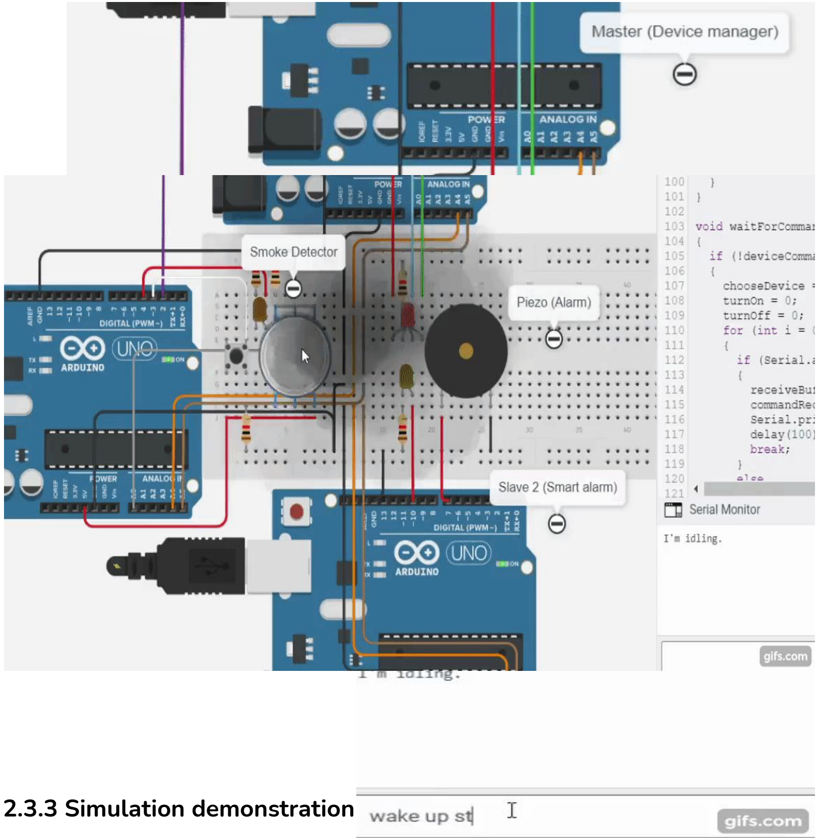


Fig. 2.3.2 Device sequence diagram



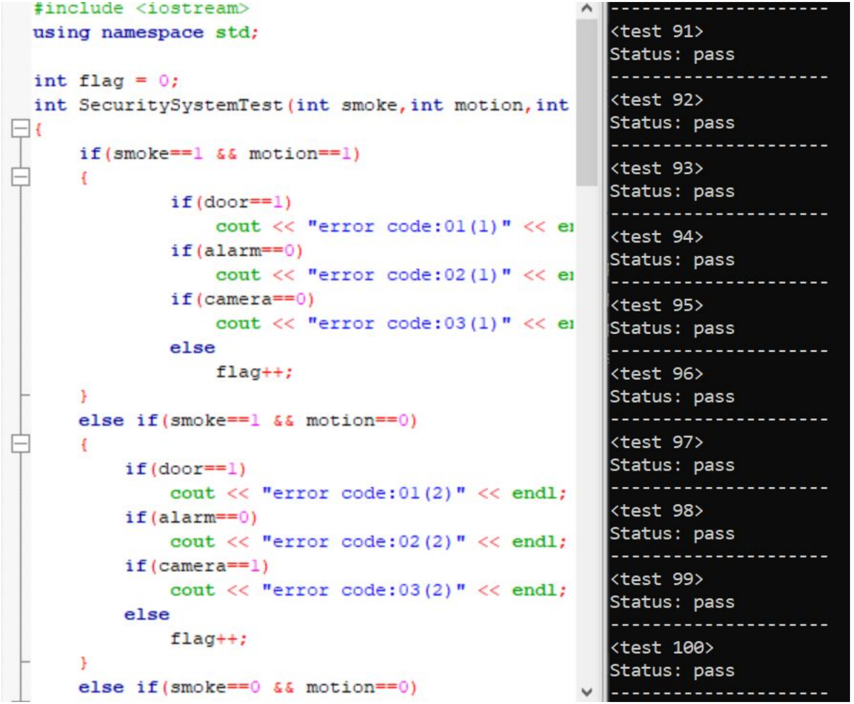
```
  }
}

void waitForComma
{
  if (!deviceComma
  {
    chooseDevice =
    turnOn = 0;
    turnOff = 0;
    for (int i = (
    {
      if (Serial.a
      {
        receiveBu
        commandRec
        Serial.pr
        delay(100)
        break;
      }
      else
```

```
I'm idling.
```

wake up st

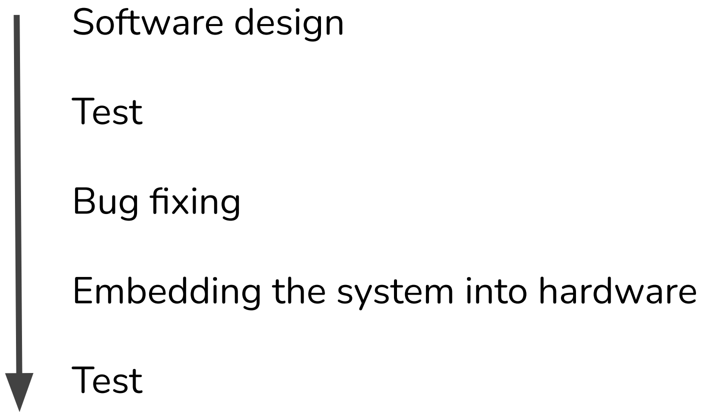
Fig. 2.3.3 Simulation demonstration

Security System



```
#include <iostream>
using namespace std;

int flag = 0;
int SecuritySystemTest(int smoke,int motion,int
{
    if(smoke==1 && motion==1)
    {
            if(door==1)
                cout << "error code:01(1)" << en
            if(alarm==0)
                cout << "error code:02(1)" << en
            if(camera==0)
                cout << "error code:03(1)" << en
            else
                flag++;
    }
    else if(smoke==1 && motion==0)
    {
        if(door==1)
            cout << "error code:01(2)" << endl;
        if(alarm==0)
            cout << "error code:02(2)" << endl;
        if(camera==1)
            cout << "error code:03(2)" << endl;
        else
            flag++;
    }
    else if(smoke==0 && motion==0)
```

```
--------------------
<test 91>
Status: pass
--------------------
<test 92>
Status: pass
--------------------
<test 93>
Status: pass
--------------------
<test 94>
Status: pass
--------------------
<test 95>
Status: pass
--------------------
<test 96>
Status: pass
--------------------
<test 97>
Status: pass
--------------------
<test 98>
Status: pass
--------------------
<test 99>
Status: pass
--------------------
<test 100>
Status: pass
--------------------
```



Software design

Test

Bug fixing

Embedding the system into hardware

Test

Fig. 3.1.2 test code

Security System

Code & implementations

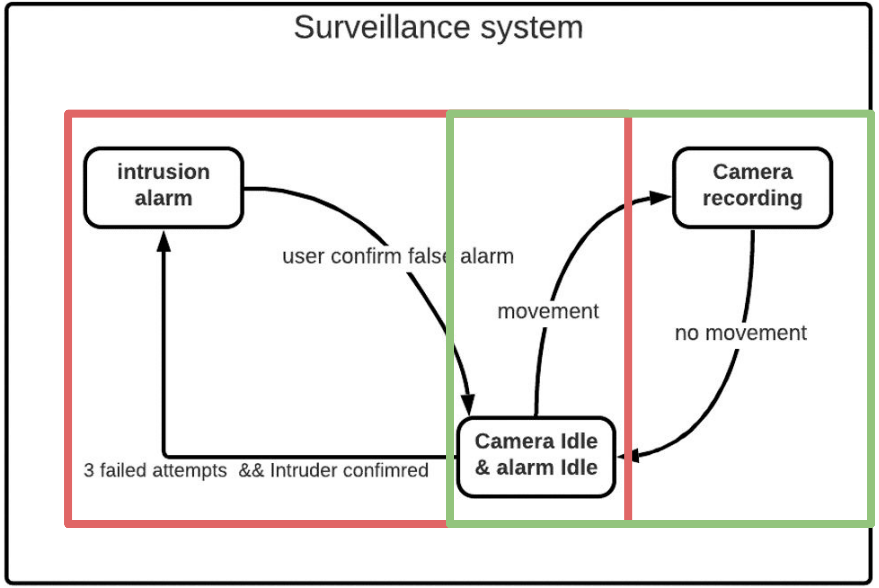
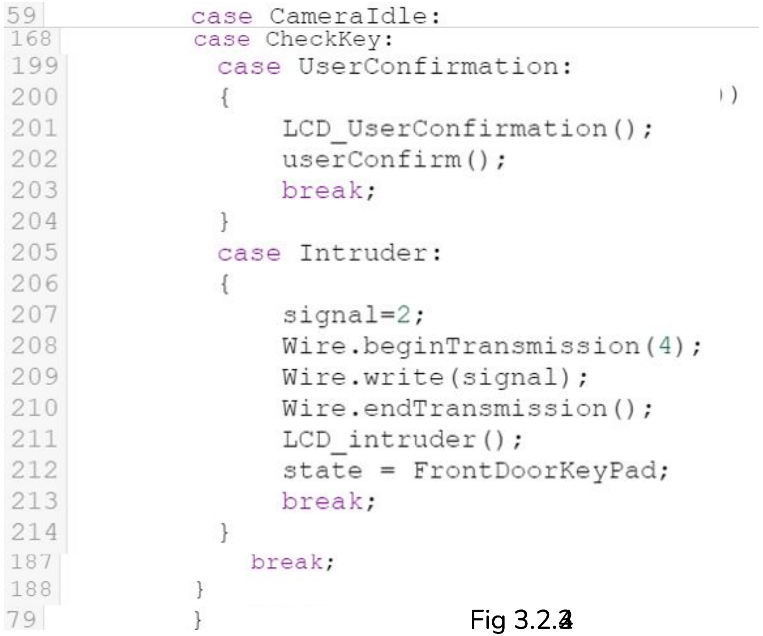


Fig. 3.2.1 state machine diagram - surveillance system



```
    case CameraIdle:
    case CheckKey:
      case UserConfirmation:
      {
            LCD_UserConfirmation();
            userConfirm();
            break;
      }
      case Intruder:
      {
            signal=2;
            Wire.beginTransmission(4);
            Wire.write(signal);
            Wire.endTransmission();
            LCD_intruder();
            state = FrontDoorKeyPad;
            break;
      }
        break;
    }
    }
```

Fig 3.2.2

Security System

Code & implementations

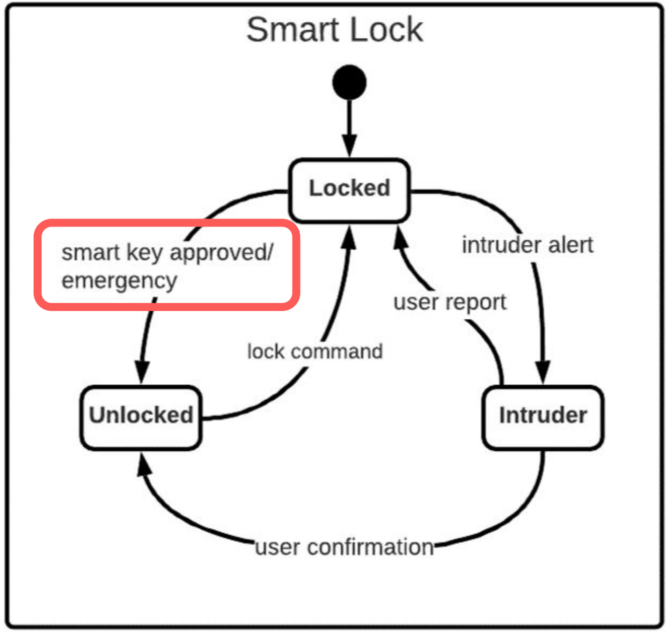
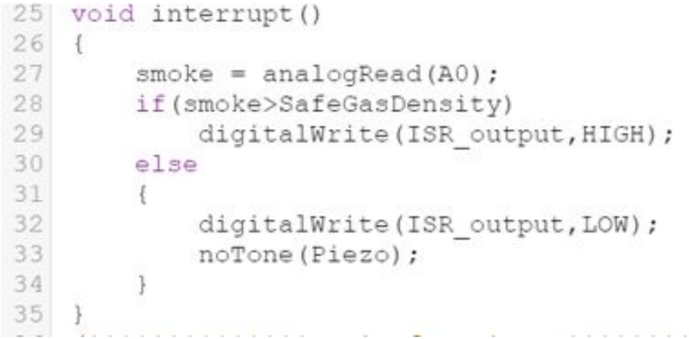
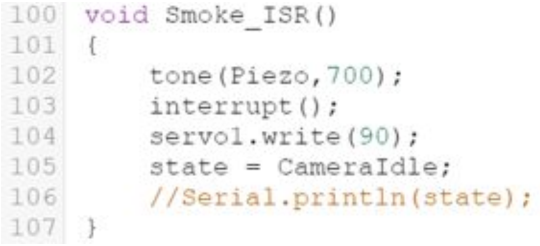


Fig. 3.3.1 state machine diagram - smart lock



```
void interrupt()
{
    smoke = analogRead(A0);
    if(smoke>SafeGasDensity)
        digitalWrite(ISR_output,HIGH);
    else
    {
        digitalWrite(ISR_output,LOW);
        noTone(Piezo);
    }
}
```

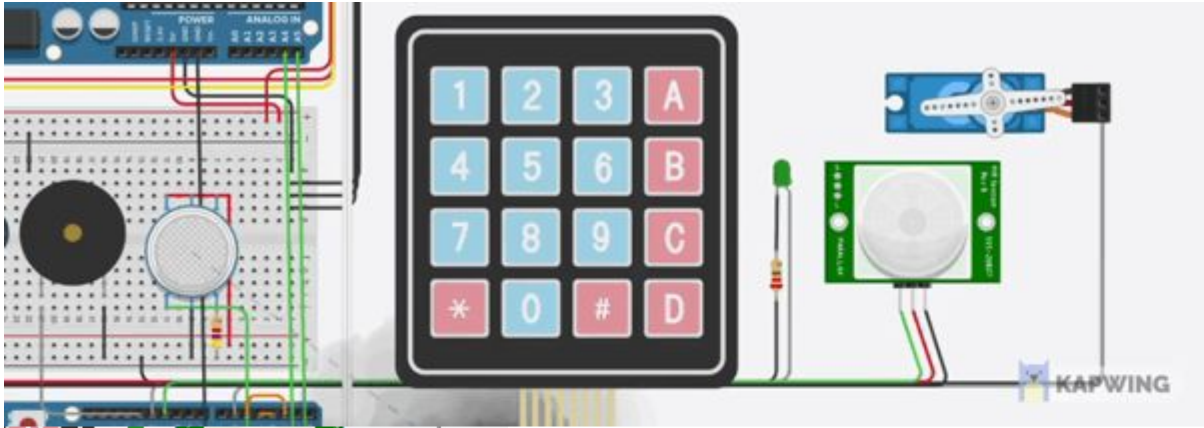


```
void Smoke_ISR()
{
    tone(Piezo,700);
    interrupt();
    servo1.write(90);
    state = CameraIdle;
    //Serial.println(state);
}
```

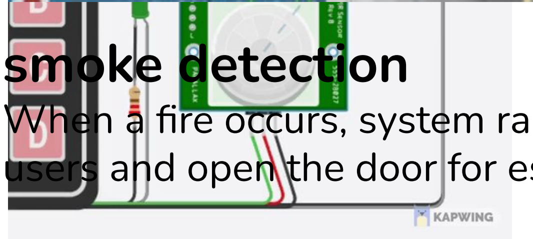
Fig. 2.2.2

Security System

Code & implementations



y usage



smoke detection

When a fire occurs, system raises alarm to warn users and open the door for escape.

Smart Lighting

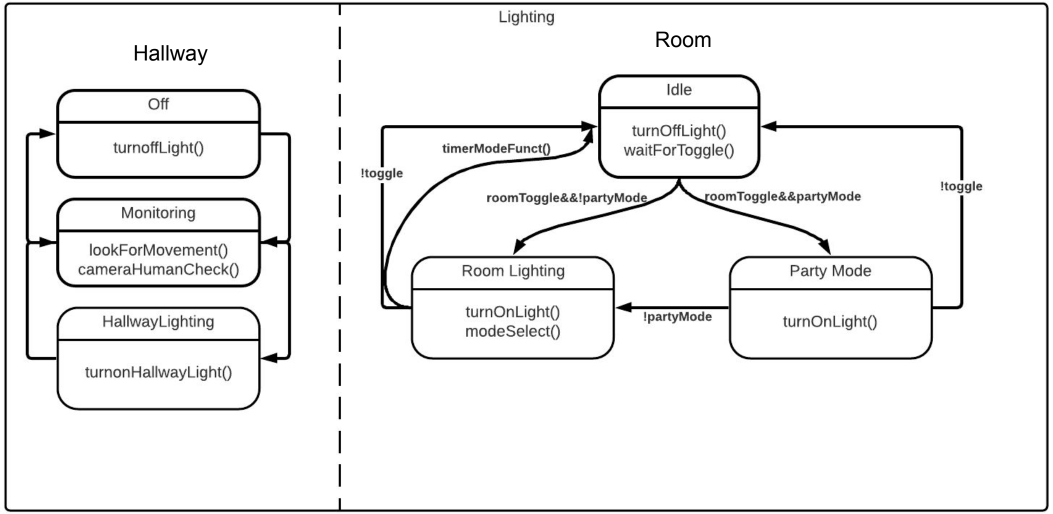
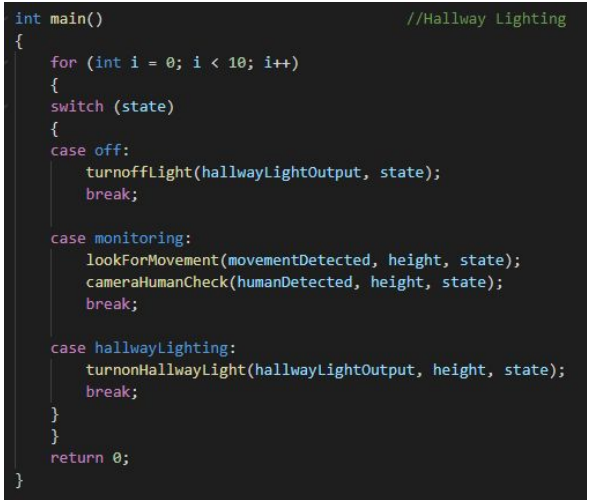


Fig. 4.1.1 State machine diagram of Smart Lighting



```
int main()                                    //Hallway Lighting
{
    for (int i = 0; i < 10; i++)
    {
    switch (state)
    {
    case off:
        turnoffLight(hallwayLightOutput, state);
        break;

    case monitoring:
        lookForMovement(movementDetected, height, state);
        cameraHumanCheck(humanDetected, height, state);
        break;

    case hallwayLighting:
        turnonHallwayLight(hallwayLightOutput, height, state);
        break;
    }
    }
    return 0;
}
```

Fig. 4.1.2 Main program code for Hallway Lighting

Smart Lighting

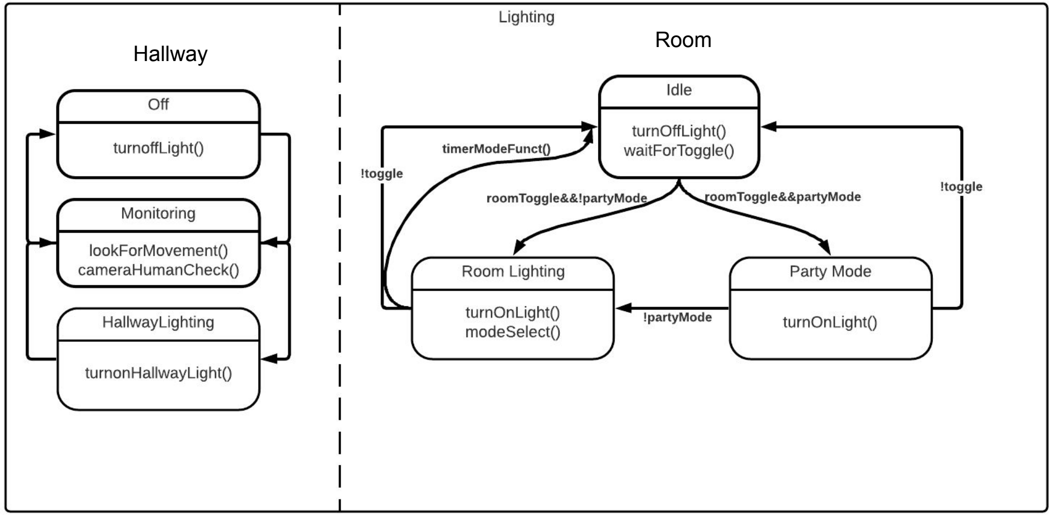
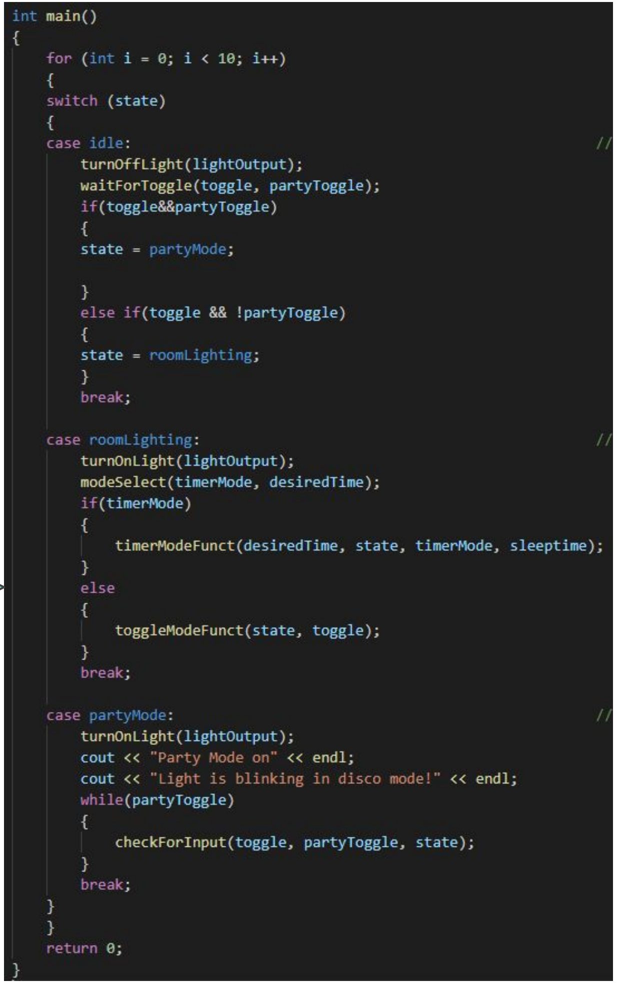


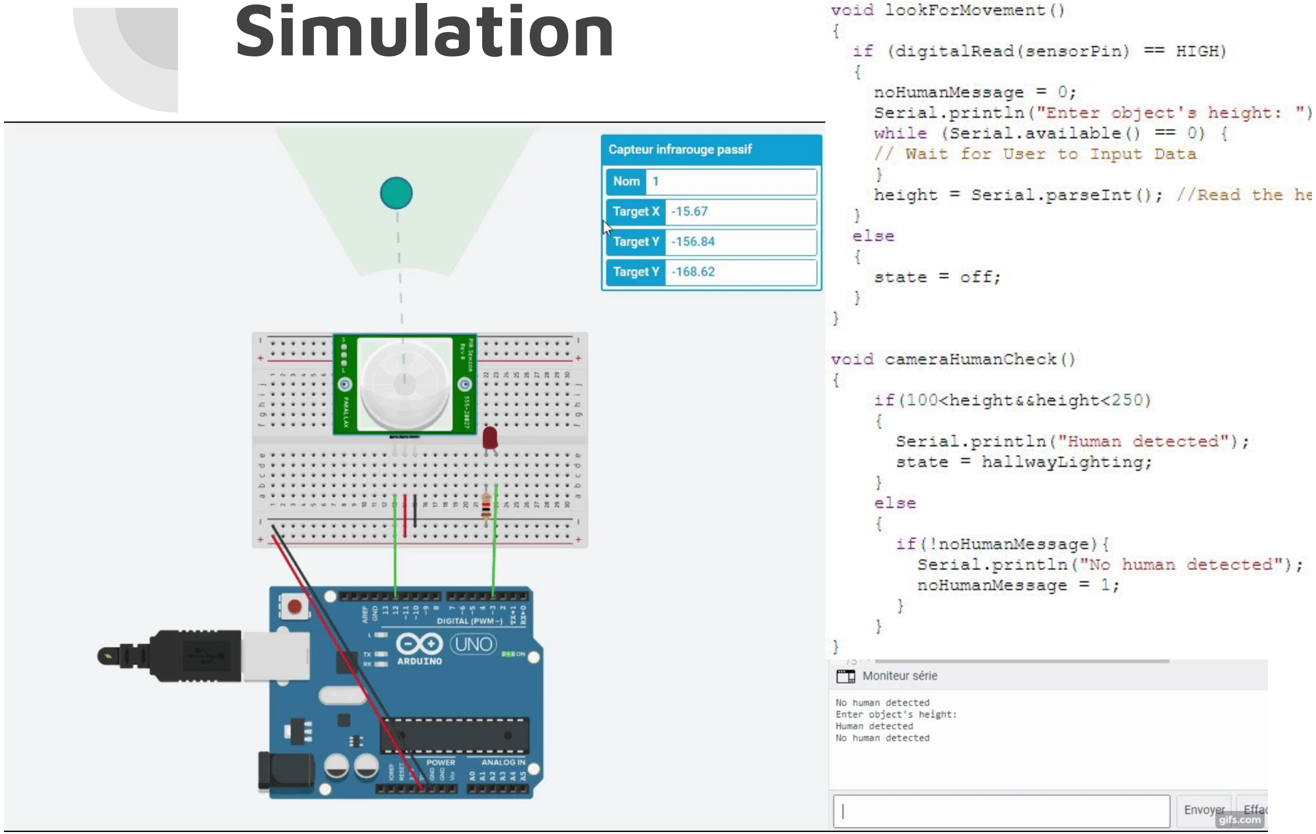
Fig. 4.2.1 State machine diagram of Smart Lighting



```
int main()
{
    for (int i = 0; i < 10; i++)
    {
    switch (state)
    {
    case idle:                                                    //
        turnOffLight(lightOutput);
        waitForToggle(toggle, partyToggle);
        if(toggle&&partyToggle)
        {
        state = partyMode;

        }
        else if(toggle && !partyToggle)
        {
        state = roomLighting;
        }
        break;

    case roomLighting:                                            //
        turnOnLight(lightOutput);
        modeSelect(timerMode, desiredTime);
        if(timerMode)
        {
            timerModeFunct(desiredTime, state, timerMode, sleeptime);
        }
        else
        {
            toggleModeFunct(state, toggle);
        }
        break;

    case partyMode:                                               //
        turnOnLight(lightOutput);
        cout << "Party Mode on" << endl;
        cout << "Light is blinking in disco mode!" << endl;
        while(partyToggle)
        {
            checkForInput(toggle, partyToggle, state);
        }
        break;
    }
    }
    return 0;
}
```

Fig. 4.1.2 Main program code for Room Lighting

Simulation



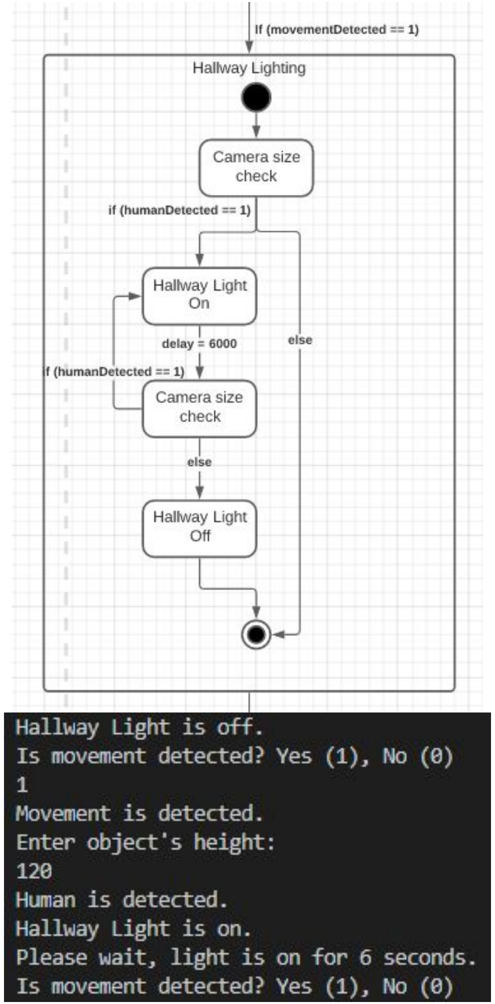
Capteur infrarouge passif

Nom	1
Target X	-15.67
Target Y	-156.84
Target Y	-168.62

```
void lookForMovement()
{
  if (digitalRead(sensorPin) == HIGH)
  {
    noHumanMessage = 0;
    Serial.println("Enter object's height: ")
    while (Serial.available() == 0) {
    // Wait for User to Input Data
    }
    height = Serial.parseInt(); //Read the he
  }
  else
  {
    state = off;
  }
}

void cameraHumanCheck()
{
    if(100<height&&height<250)
    {
      Serial.println("Human detected");
      state = hallwayLighting;
    }
    else
    {
      if(!noHumanMessage){
        Serial.println("No human detected");
        noHumanMessage = 1;
      }
    }
}
```

```
No human detected
Enter object's height:
Human detected
No human detected
```

Fig. 4.3.1 Simulation of Hallway Lighting



```
Hallway Light is off.
Is movement detected? Yes (1), No (0)
1
Movement is detected.
Enter object's height:
120
Human is detected.
Hallway Light is on.
Please wait, light is on for 6 seconds.
Is movement detected? Yes (1), No (0)
```

Fig. 4.3.2 Activity diagram & Compiled result

Grid-Operation

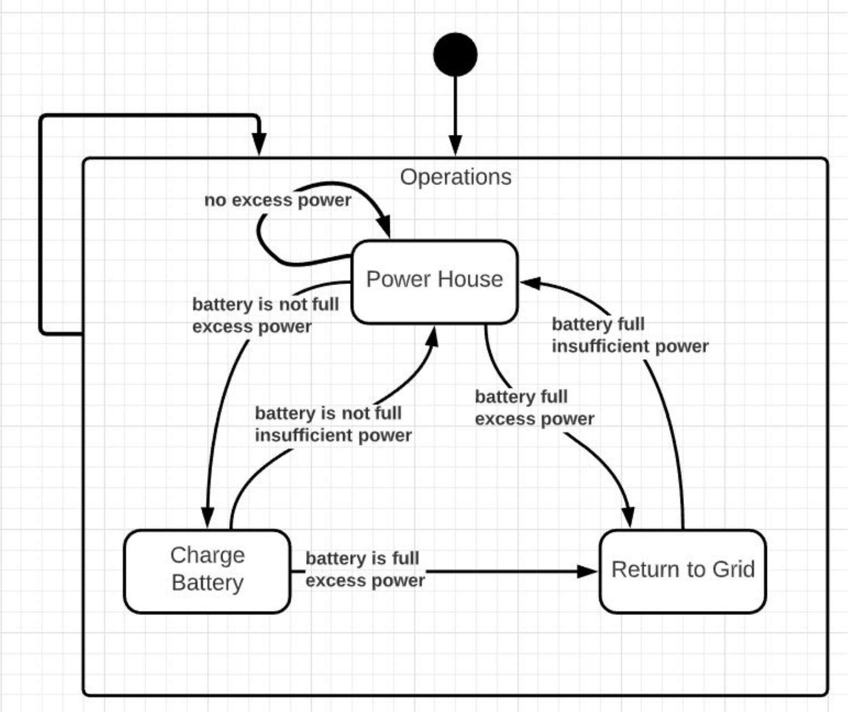
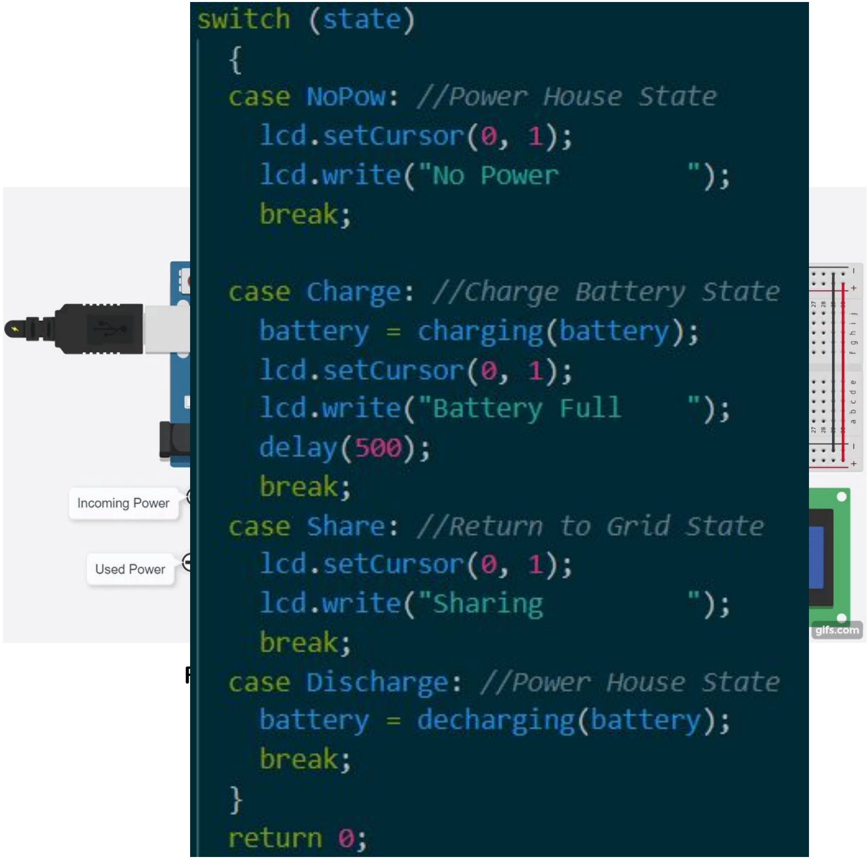


Fig. 5.1.1 State Machine diagram



```
switch (state)
  {
  case NoPow: //Power House State
    lcd.setCursor(0, 1);
    lcd.write("No Power        ");
    break;

  case Charge: //Charge Battery State
    battery = charging(battery);
    lcd.setCursor(0, 1);
    lcd.write("Battery Full    ");
    delay(500);
    break;
  case Share: //Return to Grid State
    lcd.setCursor(0, 1);
    lcd.write("Sharing         ");
    break;
  case Discharge: //Power House State
    battery = decharging(battery);
    break;
  }
  return 0;
```

Fig. 5.1.2 Switch case based on Fig. 5.1.1

Grid-Monitoring

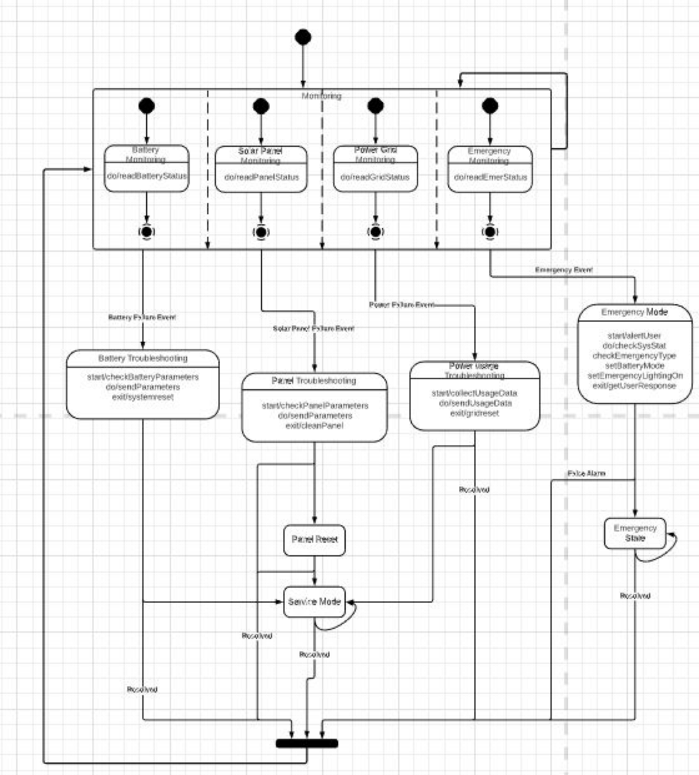
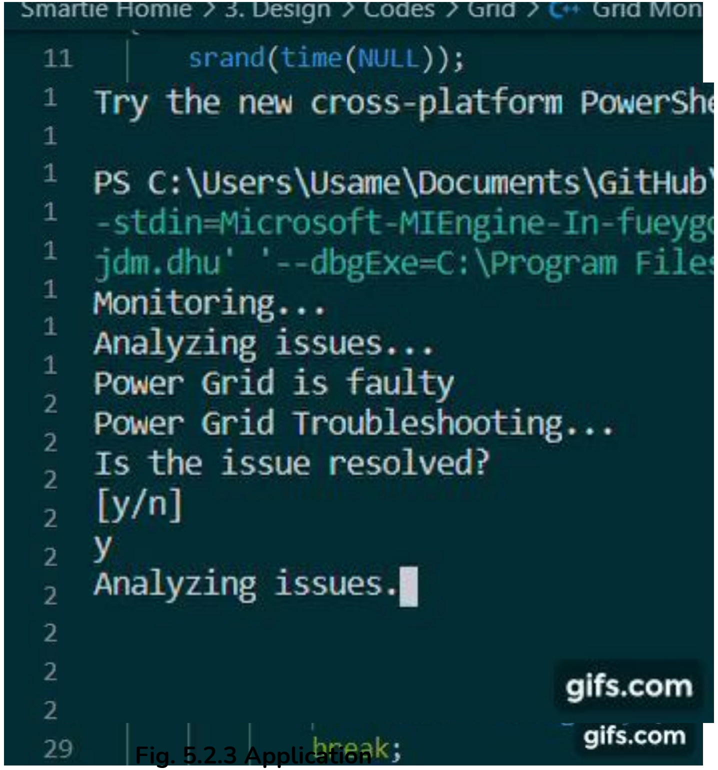


Fig. 5.2.1 State Machine diagram



```
        srand(time(NULL));
```

```
Try the new cross-platform PowerShe

PS C:\Users\Usame\Documents\GitHub
-stdin=Microsoft-MIEngine-In-fueyg
jdm.dhu' '--dbgExe=C:\Program File
Monitoring...
Analyzing issues...
Power Grid is faulty
Power Grid Troubleshooting...
Is the issue resolved?
[y/n]
y
Analyzing issues.
```

```
            break;
```

Fig. 5.2.3 Application

Fig. 5.2.2 Switch case based on Fig 4.2.1

Smart Entertainment
_Activity Diagram & State Machine Diagram

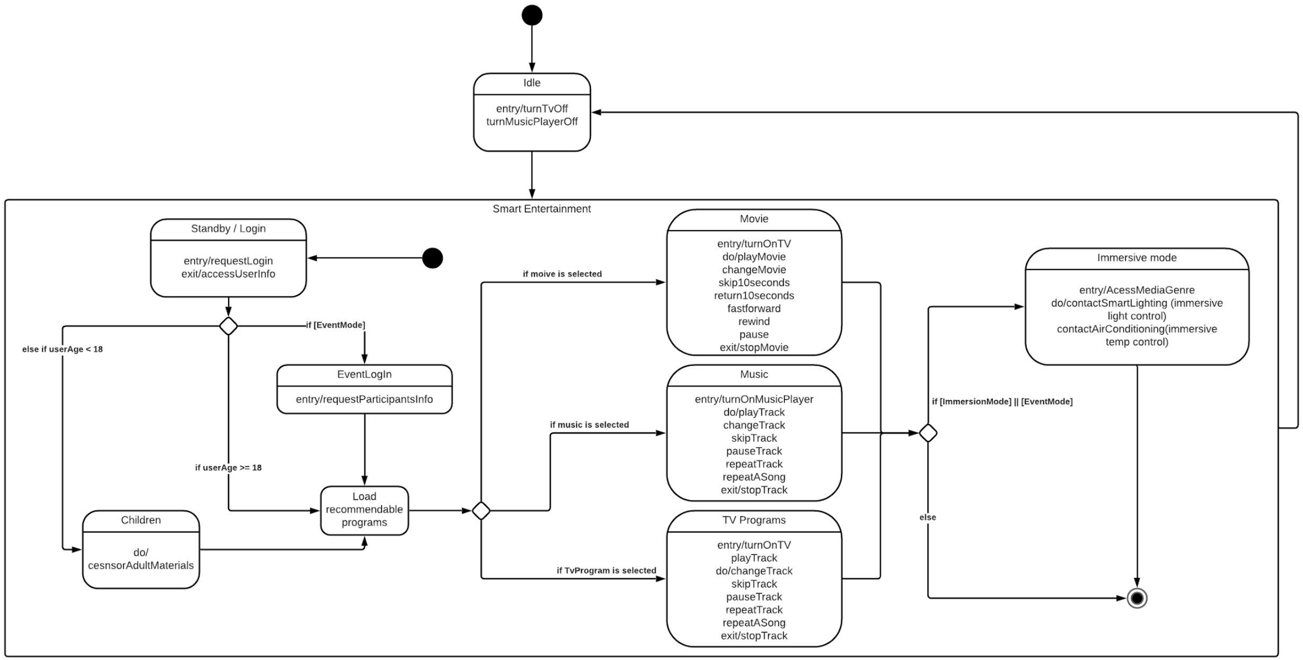


Fig. 6.1.1 State Macine Diagram

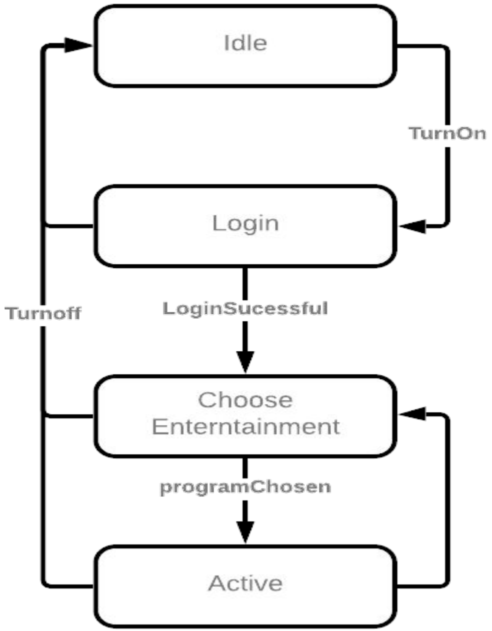
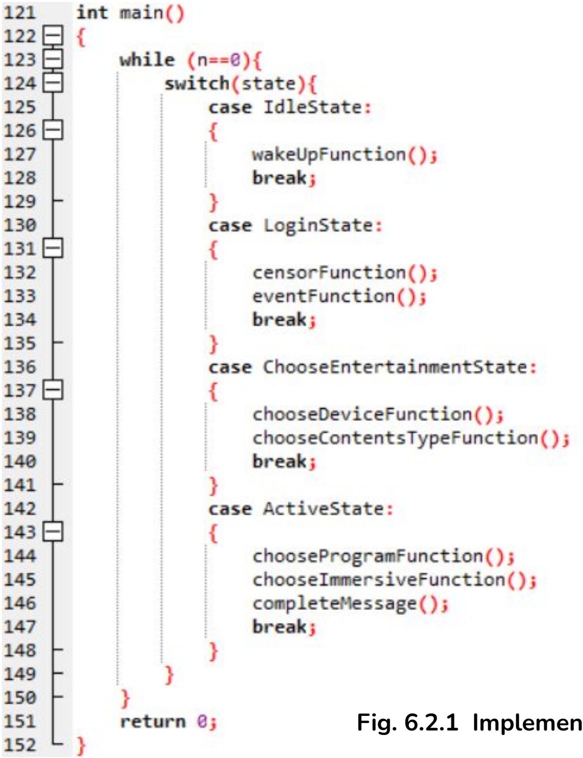


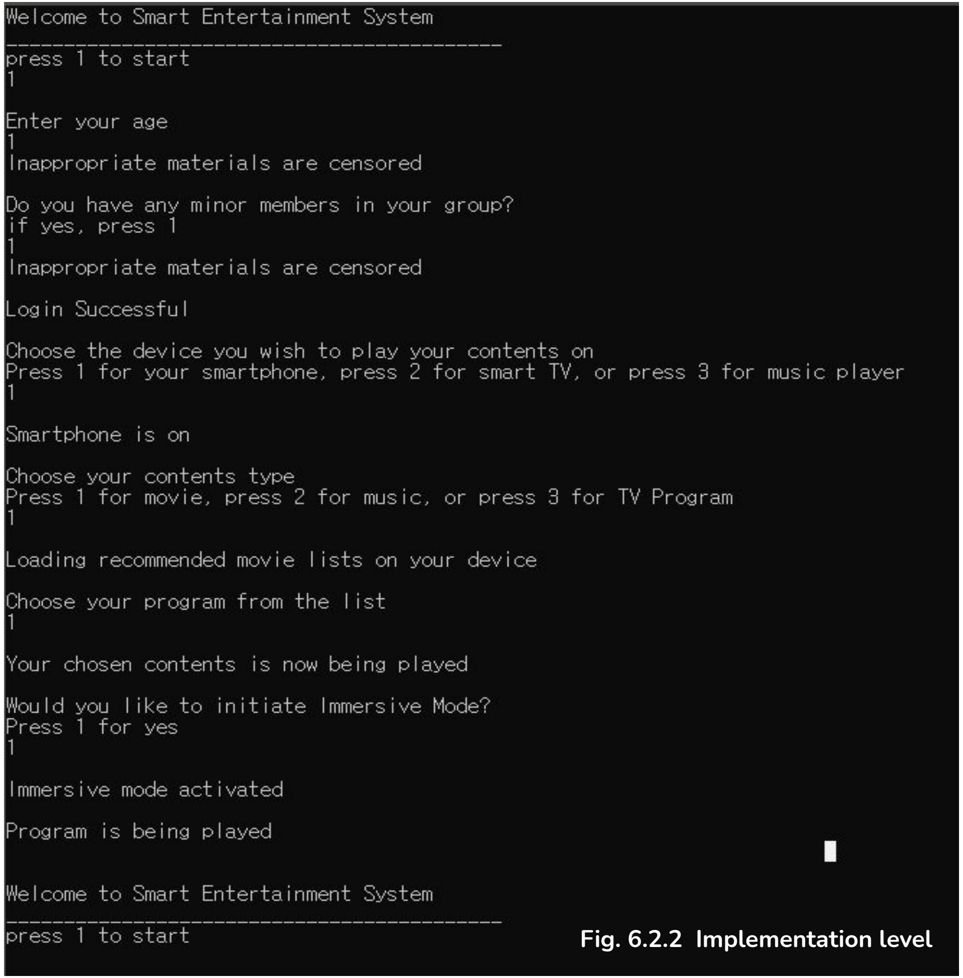
Fig. 6.1.2 Revised State Machine Diagram

Smart Entertainment _Implementation Code



```
int main()
{
    while (n==0){
        switch(state){
            case IdleState:
            {
                wakeUpFunction();
                break;
            }
            case LoginState:
            {
                censorFunction();
                eventFunction();
                break;
            }
            case ChooseEntertainmentState:
            {
                chooseDeviceFunction();
                chooseContentsTypeFunction();
                break;
            }
            case ActiveState:
            {
                chooseProgramFunction();
                chooseImmersiveFunction();
                completeMessage();
                break;
            }
        }
    }
    return 0;
}
```

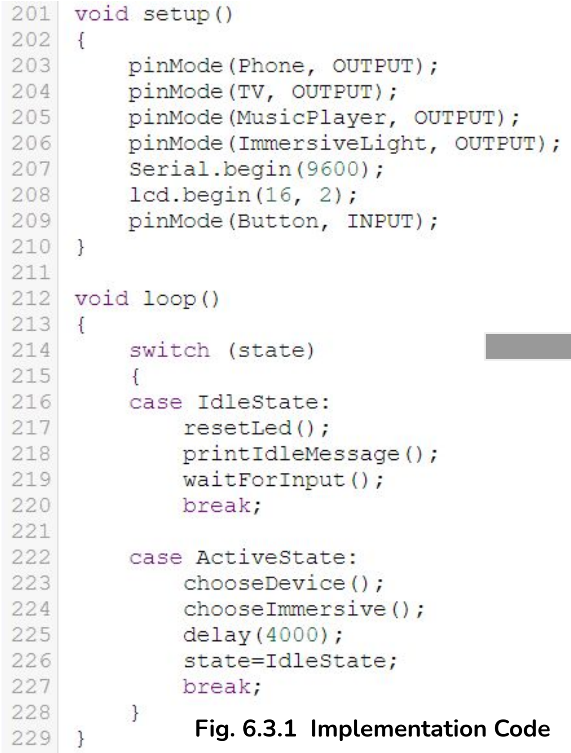
Fig. 6.2.1 Implementation Code



```
Welcome to Smart Entertainment System
-------------------------------------------
press 1 to start
1

Enter your age
1
Inappropriate materials are censored

Do you have any minor members in your group?
if yes, press 1
1
Inappropriate materials are censored

Login Successful

Choose the device you wish to play your contents on
Press 1 for your smartphone, press 2 for smart TV, or press 3 for music player
1

Smartphone is on

Choose your contents type
Press 1 for movie, press 2 for music, or press 3 for TV Program
1

Loading recommended movie lists on your device

Choose your program from the list
1

Your chosen contents is now being played

Would you like to initiate Immersive Mode?
Press 1 for yes
1

Immersive mode activated

Program is being played


Welcome to Smart Entertainment System
-------------------------------------------
press 1 to start
```

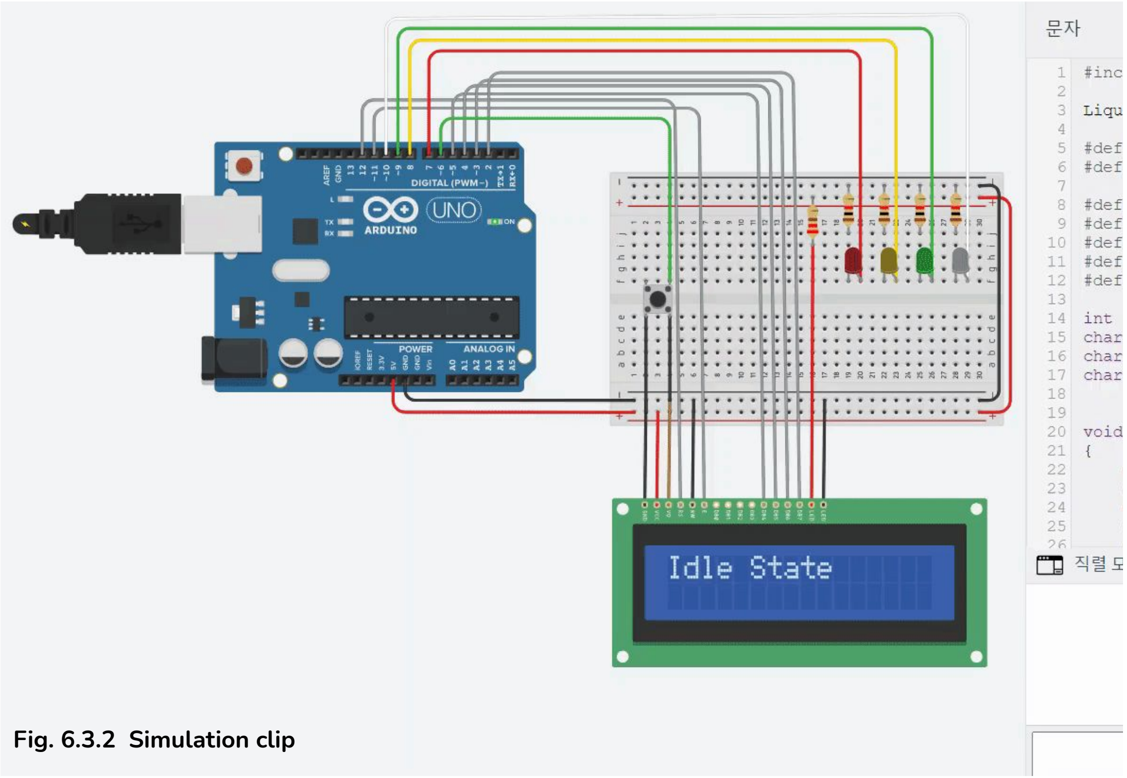
Fig. 6.2.2 Implementation level

Smart Entertainment
_Arduino Design, Codes and Simulation



```
void setup()
{
    pinMode(Phone, OUTPUT);
    pinMode(TV, OUTPUT);
    pinMode(MusicPlayer, OUTPUT);
    pinMode(ImmersiveLight, OUTPUT);
    Serial.begin(9600);
    lcd.begin(16, 2);
    pinMode(Button, INPUT);
}

void loop()
{
    switch (state)
    {
    case IdleState:
        resetLed();
        printIdleMessage();
        waitForInput();
        break;

    case ActiveState:
        chooseDevice();
        chooseImmersive();
        delay(4000);
        state=IdleState;
        break;
    }
}
```

Fig. 6.3.1 Implementation Code



```
#inc

Liqu

#def
#def

#def
#def
#def
#def
#def

int
char
char
char


void
{
```

Fig. 6.3.2 Simulation clip

Thank you for listening!

Many more features available to be discovered!

Reference

[1]Maker Pro, Arduino UNO. 2015.