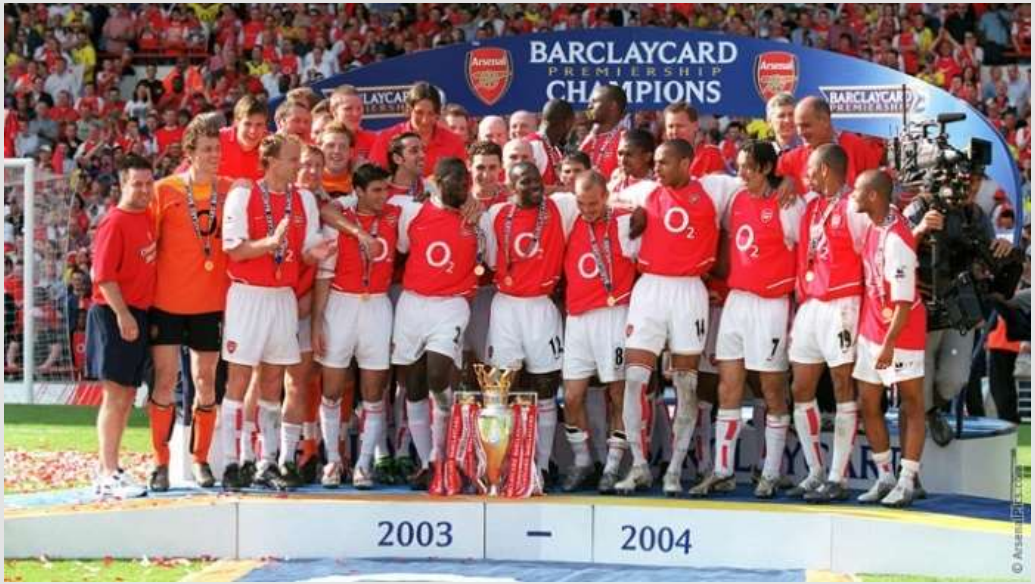
Rescue Robot

“The Invincibles”

- *Yahia Mesharaf*
- *Brian Visas*
- *Enkeledi Mema*
- *Hermann Anguiga*
- *Aluko Tunde Oluwayemi*

Hint: that’s just a rough draft of the presentation not the one we will really present

Why the Invincibles?



Arsenal 2004

Diagrams on System Architecture:

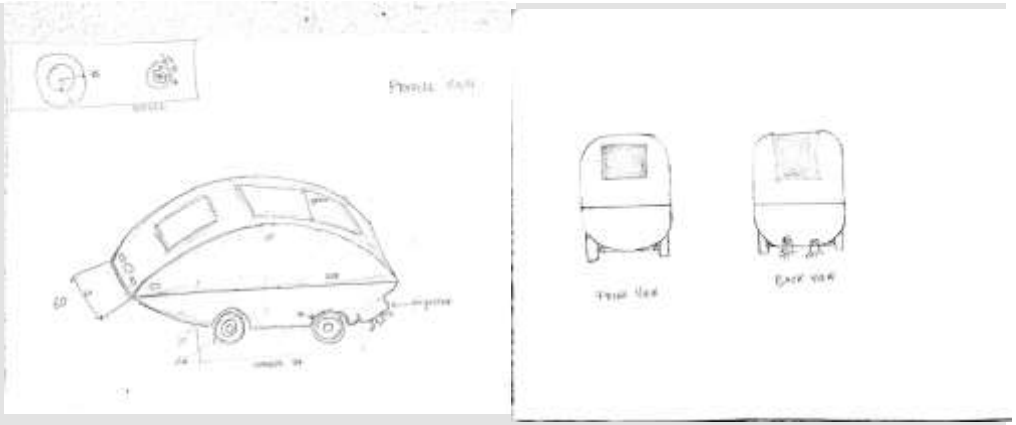
Use case D. →→ Requirement D. →→ Block Diagram

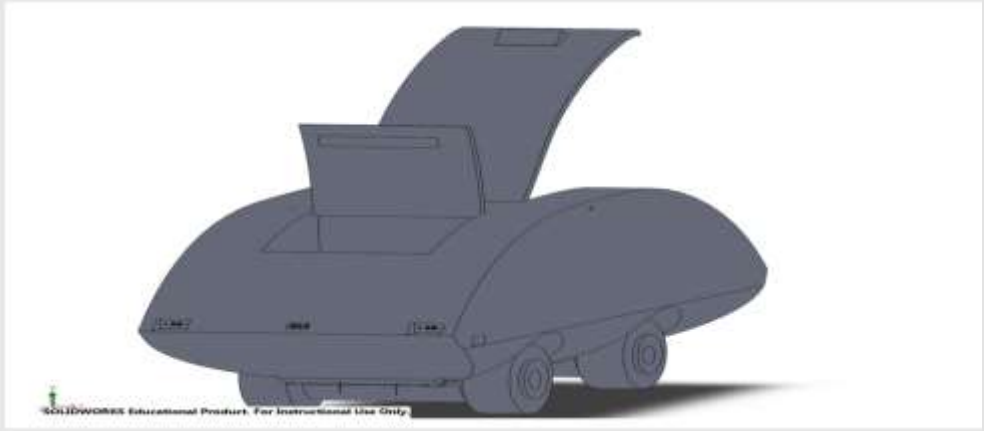
Subsystem Realization:

Relation between Subsystems as well as algorithms

3D Model

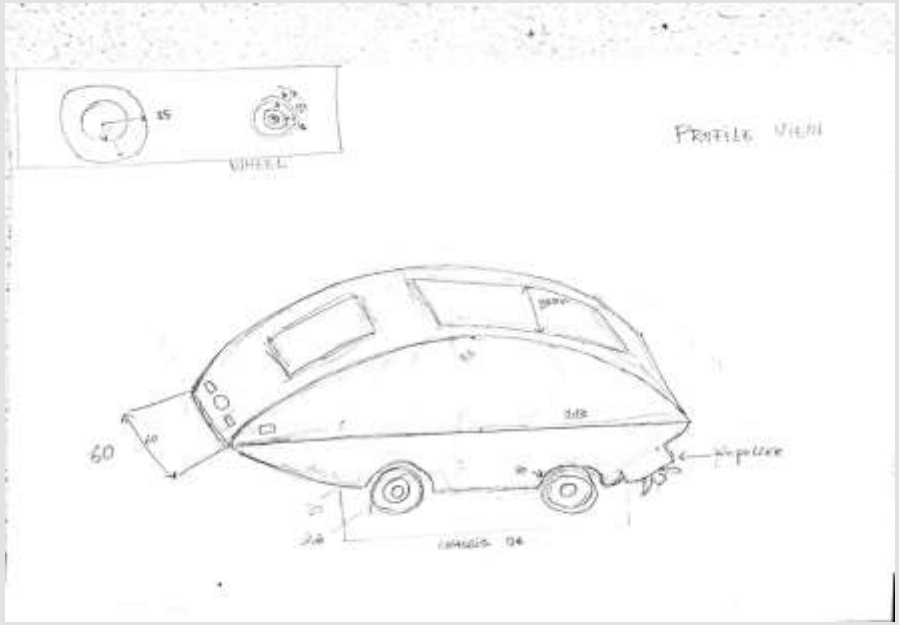
1. *Conception*
2. *Layout*
    - *General structure*
1. *Affordances and Signifiers*
2. *General Scenario*





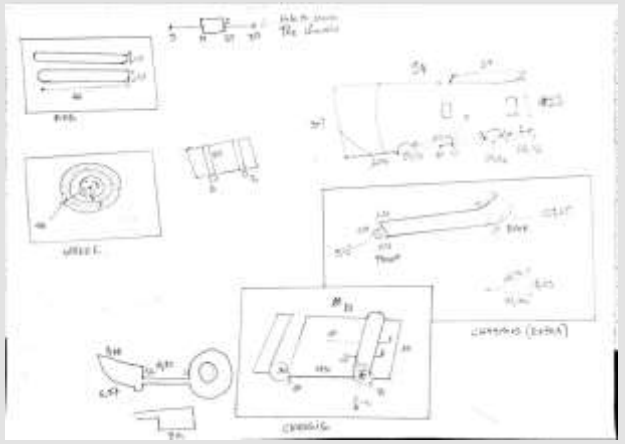
Conception

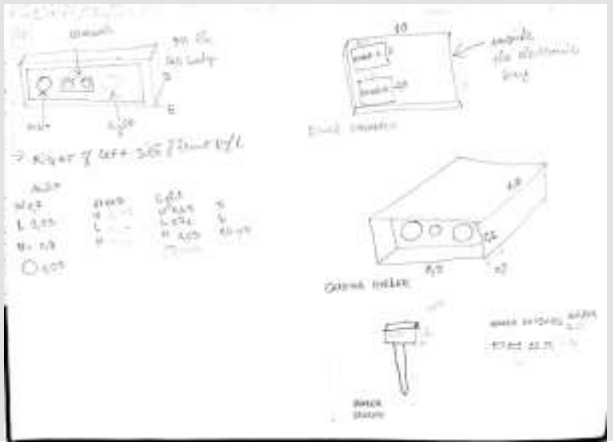
Why this shape?



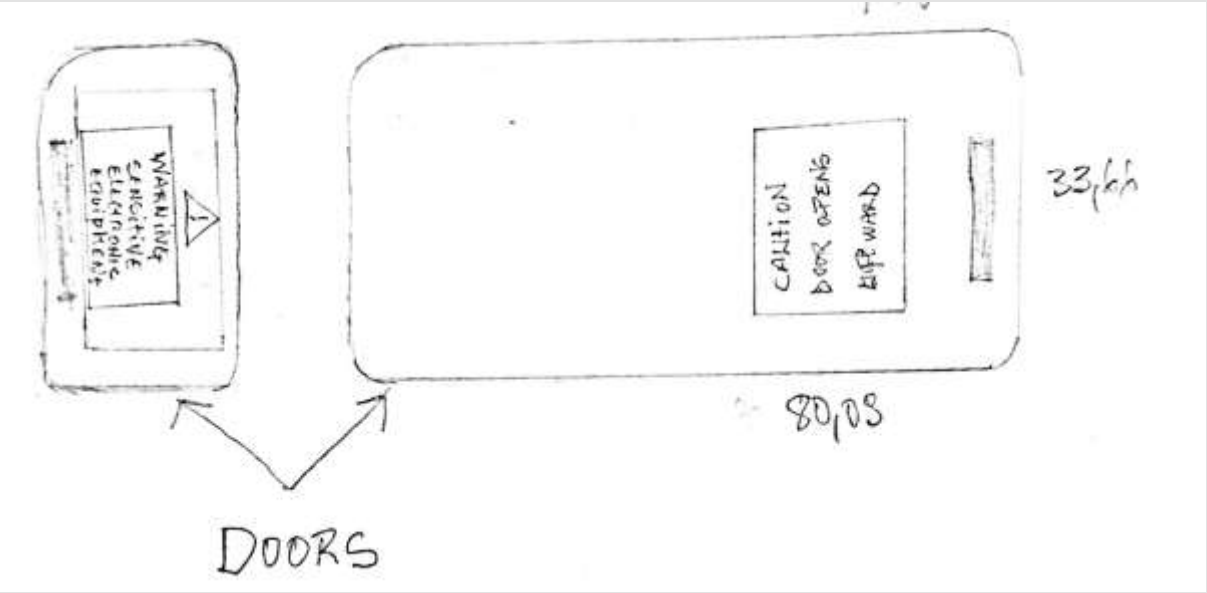
Layout

1. *The Body*
2. *The Chassis*
3. *The Sensors and Boards holders*



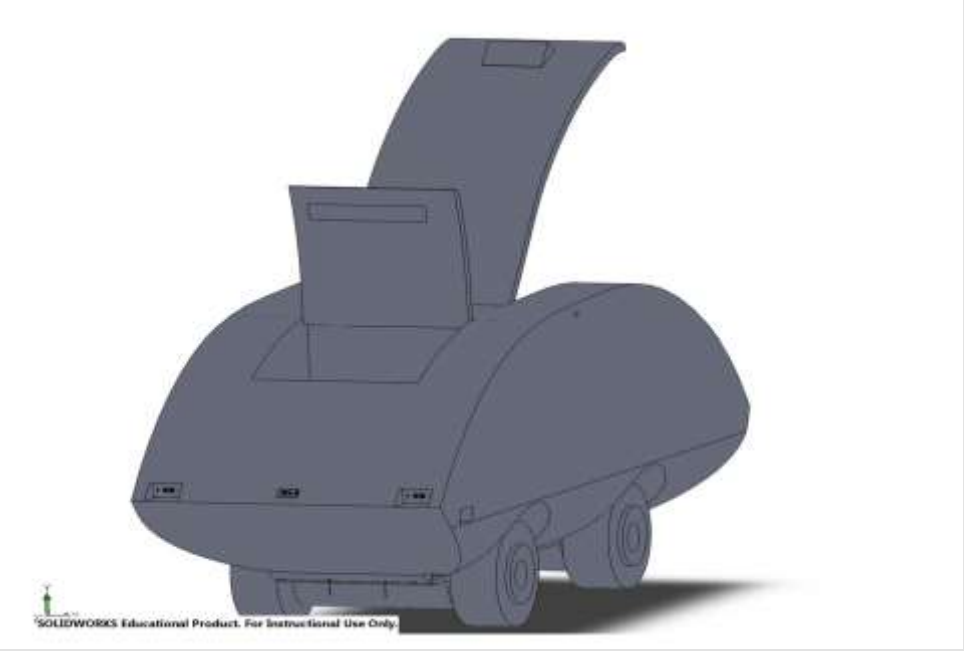


Affordances and Signifiers



General Scenario

The Final 3D Model

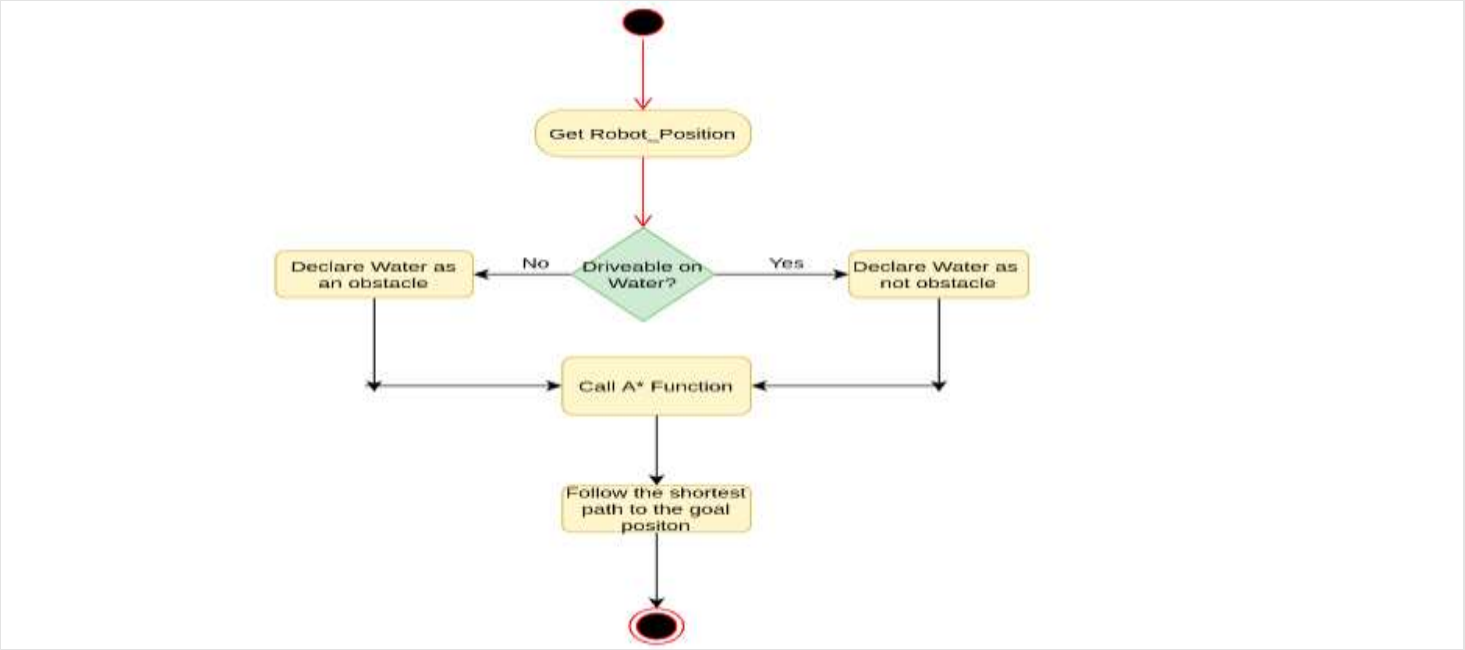


Simulations:

- Videos
- Softwares

A Look into the Future ..

Driving Algorithm

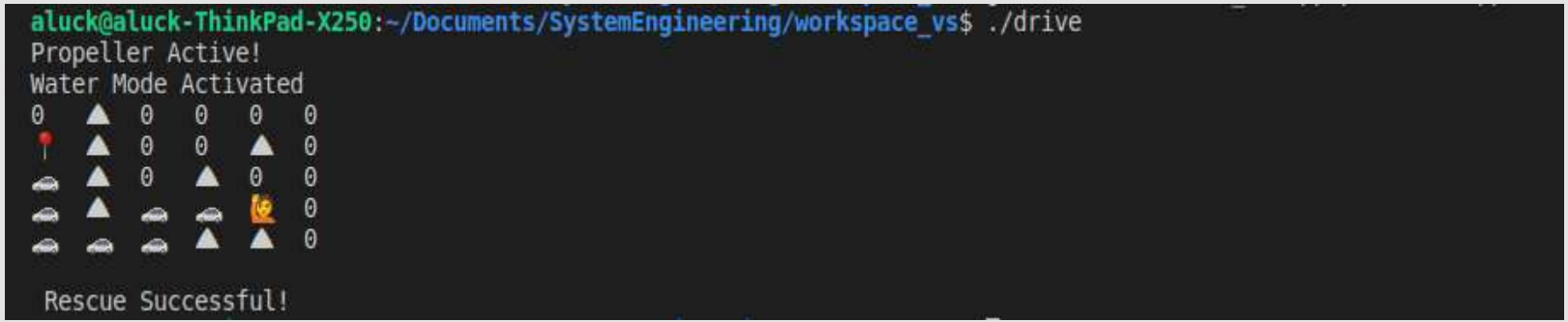


A* Shortest Path Algorithm Implementation

```
119  vector<vector<State>> Search(vector<vector<State>> environment, int init[2], int goal[2]) {
120    // Create the vector of open nodes.
121    vector<vector<int>> open {};
122
123    // Initialize the starting node.
124    int x = init[0];
125    int y = init[1];
126    int g = 0; //g-value
127    int h = Heuristic(x, y, goal[0],goal[1]);
128    AddToOpen(x, y, g, h, open, environment);
129
130    while (open.size() > 0) {
131      // Get the next node
132      CellSort(&open);
133      auto current = open.back();
134      open.pop_back();
135      x = current[0];
136      y = current[1];
137      environment[x][y] = State::sPath;
138
139      // Check if we're done.
140      if (x == goal[0] && y == goal[1]) {
141        environment[init[0]][init[1]] = State::sInit;
142        environment[goal[0]][goal[1]] = State::sObject;
143        return environment;
144      }
145
146      // If we're not done, expand search to current node's neighbors.
147      ExpandNeighbors(current, goal, open, environment);
148    }
```

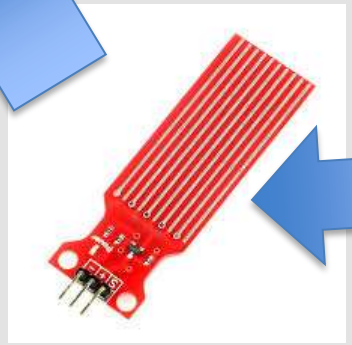
Test Case Result

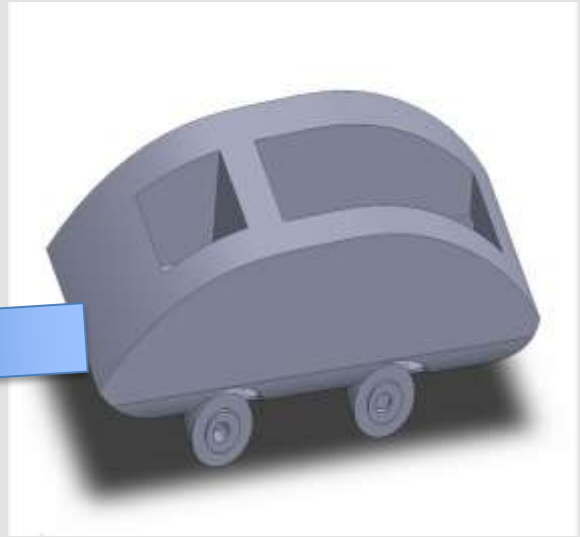
The test result shows how our driving actually avoid obstacles and choose the shortest path to goal destination.



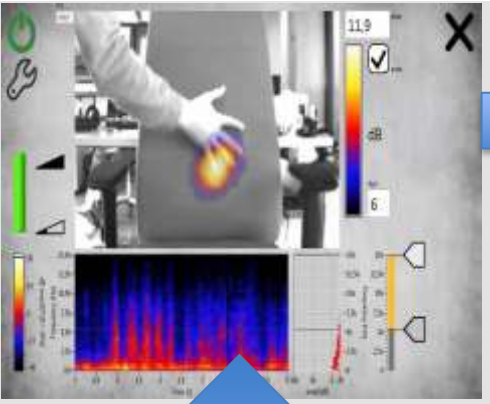
Implementation

- Video for hardware simulation.

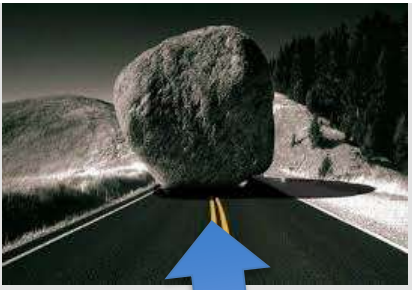




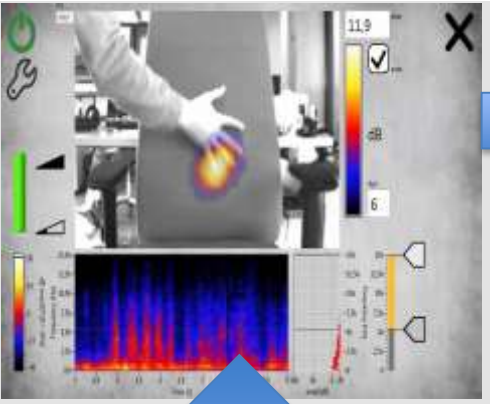
```
#include <iostream>

void Drive() {

}

bool checkRescueRobot() {
        if(driveAbleOnWature)
                Return True;
        else return False;
}

if (checkRescue == True) {
        std::vector<RouteModel::Node> obstacle;
        obstacle.emplace_back("Water");
}

float next_path = Astar().path;

int main(){
        Drive();
}
```



```
int svs_on();
int main_camera_on();
int main_camera_off();
int night_vision_on();
int night_vision_off();
int infrared_on();
int infrared_off();
int connection_with_main_controller();
int data_processing();
int video_stream();

void main()
{
  int brightness, resolution, x, y;

  while(svs_on()){
  main_camera_on();

  if(brightness <x){
  night_vision_on();
  main_camera_off();
  }

  if(resolution <y){
  infrared_on;
  main_camera_off();
  }

  video_stream();
  connection_with_main_controller();
```



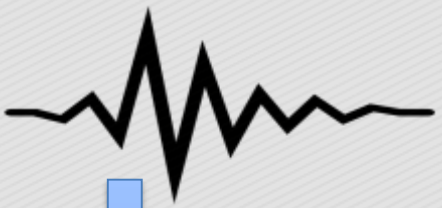
```
int check_ObstracCars()
{

  int Distance
  {
    if ( FrontSensor() < Distance && RightSensor () >Distance && LeftSensor
    ()>Distance) // obstacle infront of all 3 sides
    {
    turn_right ();

    }


    else if (FrontSensor() >Distance && RightSensor () >Distance && LeftSensor
    ()>Distance) // obstacle on right and front sides
    {
    turn_left (); // turn left side
    }


    else if (FrontSensor() >Distance && RightSensor () >Distance && LeftSensor
    ()>Distance) // obstacle on left and front sides
    {
    turn_right (); // turn right side
    }


    else if (FrontSensor() >Distance && RightSensor () >Distance && LeftSensor
    ()>Distance) // obstacle on front sides
    {
    turn_right (); // then turn right
    }


    else if (FrontSensor() >Distance && RightSensor () >Distance && LeftSensor
```

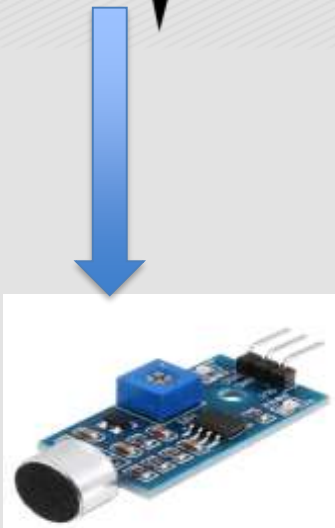


```
int svs_on();
int main_camera_on();
int main_camera_off();
int night_vision_on();
int night_vision_off();
int infrared_on();
int infrared_off();
int connection_with_main_controller();
int data_processing();
int video_stream();

void main()
{
  int brightness, resolution, x, y;

  while(svs_on()){
  main_camera_on();

  if(brightness <x){
  night_vision_on();
  main_camera_off();
  }

  if(resolution <y){
  infrared_on;
  main_camera_off();
  }

  video_stream();
  connection_with_main_controller();
```

```
int Rescue()
{
  // example

  while(rescue='1') //Rescue object is nearby
  {
    if(door_closed)//The door or cover is locked
    {

      if(front_sensor='1') //Sensor detect the target
      {
        open_door();
        if(open_door='1');//Door open successfully
        {
            Give_help ();
            {

            if(give_help='1')//object is rescued
             printf("Help received");
             else()
             {

             printf("Help not received");
             Manual_Help_On(); //The help is controlled manually
             }
        }
        else {

        printf("Door could not open automaticaly");
        Manual_Mode_ON();//The door is open manually

    }
      }
      else
      locate_target();//Sensor searching for the target
    }
```

Compile Log Debug Find Results





Audio_Detect_E.Mena.cpp | Check_Condition.cpp | Check_for_environment_Obstacles_E.Mena.cpp | Path_E.Mena.cpp | rescue_v_2_E.Mer

```
Audio_Detect()
{
    if(mic==1)//Surveillance mode on
    {
        if(Mic_on=1)//Mic turn on successfully
        {
            while(1)
            {
            printf("Watting for audio signal");
            if(Signal=true)
            {
                Record_signal_Analog();//get the  analog signal
                ADC();//Convert singnal from analog to digital
                time_sample_intervals();//check the range of signal and analyze all information
                fergency();//detetc the wave frequency
                DAC();//Convert singnal from digits to analog

            }
            else
            {
            printf("Audio signal undefined");
            Analyze_Manual_Mode_on();//The audio sample is analyzed manually
            }
            }
        }
        else
        {
            printf("Mic could not tur on autometyclly");
            Mic_On _Manual_Mode();

        }
    }
}
```