
# Addis Ababa University

## Addis Ababa Institute of Technology

### School of Electrical and Computer Engineering

**Smart Cane for Blind People**

**Kelem Negasi ATR/8467/07**

**Meles Kidane ATR/6466/07**

**Advisor: Mr. Dagmawi Demissie**

**Date of submission**: 7/1/2019

# Abstract

The purpose of this project is to solve the difficulty of blind people in moving from one place to another place and simplify their day to day activities. As we all know most blind people depend on other people's guidance and a hand held stick to avoid obstacles and walk outside. In this project we design and implement a smart device to solve and simplify the problem of identifying obstacles using sensors and computer vision solutions.

This device will include proximity sensors to sense the distance from obstacles, light sensors to sense the light conditions of the environment, moisture sensor to sense water, USB web cam and raspberry pi to achieve the desired goal. All the data gathered by the sensors and the camera is fed to the raspberry pi in order to get processed inform the user. It will also include a buzzer to notify the user that an obstacle has been detected. The device will benefit visually impaired poeople navigate to any place without the need of other people's guidance.

# Table of Contents

# Table of figures

# Chapter 1 Introduction

## 1.1 Background

Visual impairment is one of the most common disabilities worldwide. According to world health organization's report, around 15% of the world population live with some sort of impairment and around 213 million people of this statistics live with severe vision impairment. More than 90% of the world's visually impaired live in developing countries. [1]

Visually impaired people face a lot of challenges while walking outside independently. The traditional white cane that most blind people use as an additional aid doesn't do much of work in detecting an obstacle like human eye can do. Even though a lot of consumer and medical technology has made significant advancements over the past 60 years it is really unfortunate to see that the functionality of canes for the visually impaired remains limited which relies only on the user's ability to physically reach objects.

This project aims to contribute to the ongoing research and invention on the modification and modernizing the white cane for the visually impaired. It explores the capability of sensors and computer vision at detecting an obstacle to design and build a smart cane that can detect an obstacle from a distance and warn the user before they actually have to touch the obstacle physically. Not only that, but it can also give direction based on the data gathered from sensors mounted on the device at different directions.

This device mainly uses ultrasonic sensors to detect bigger obstacles, infrared sensors to detect small and closer obstacles, temperature sensors to detect fire area, photosensitive resistor or light dependent resistor to detect the possibility of getting into darker area. This device uses the rasobery pi microcontroller to compute the data and give appropriate response.

The camera based object recognition works based on openCV library to detect targe objects based on pretrained haar cascade classifier. A USB web cam is used to capture the images fee to the image recogniton system.

## 1.2 Problem Definition and motivation

Navigation is one of the common activities of human beings. To successfully travel from one place to another, we have to make sense of our environment so that we can manage to avoid obstacles and follow the best route. Our eye is one of the sensory organs our body that plays a huge role in sensing our environment using the visible light. Visual impairment makes people unable to sense the visible world through light.

Visually impaired people uses a conventional cane to navigate around and avoid obstacles by detecting using the stick. Even though this helps to simplify the problem of obstacle detection, it is limited in in its capability of detecting obstacles because it can only detect those obstacles within the physical reach of the stick itself. Due to this reason, a blind person can't farther know and detect obstacles and decide the possible route and direction he should follow to avoid the obstacle. This could be simplified if the walking stick was equipped with devices capable of detecting obstacles beforehand and generate directiions based on the kind of environment detected.

## 1.3 Objectives of the Project

### 1.3.1 Main objective

The main objectives of the project are:

- To solve the problem of navigation of visually impaired people
- To search and study an alternative way of obstacle detection
- To modify and modernize the conventional cane
- To design and build a new and smarter blind stick that has fast response and wide range of sensory data.

### 1.3.2 Specific objectives:

- To use sensors to perceive and make sense of the environment
- To implement the principles of computer vision to see and identify objects and obstacles
- To test and measure the capability of different kind of sensors in identifying and recognizing obstacles.
- To develop image recognition system using camera and openCV image recognition libraries

# Chapter 2 Literature Review and Theoretical Background

## 2.1 Literature Review

Several attempts have been made to design obstacle avoidance devices for the blind using components with limited number of applications. This section will discuss some of these attempts and their features.

A paper by Mithiles Kumar, Md Faysal Kabir and Sahadev Roy titled "Low Cost Smart Stick for Blind and Partially Sighted People" proposes an approach to overcome the major challenges faced by blind and partially blind people to cross the road. [2] This paper presents the design of new device capable of detecting large and small obstacles, fire area and darkness using different kinds of sticks

A paper proposed by Oluwaseyitan Joshua Durodola, Nathaniel Sims and Chris Uruquhart explores the design of a little bit advanced and expensive system that includes obstacle detection sensors, GPS navigation and GSM internet with Intel Atom microprocessor as its processor [3]. Even though this approach is very accurate and precise at detecting obstacles and giving direction, its prices is not affordable to every one especially to low income individuals.

Another paper written on this subject matter is by Dada Emmanuel Gbenga, Arhyel Ibrahim Shani, Adebimpe Lateef Adekunle which uses ultrasonic sensor, moisture detection system and Arduino microcontroller to assist the visually impaired on their day to day activity [4].

A Self-Energized Smart Vision Stick for Visually Impaired People was designed by Mohammad Hossain Mohammadi,Saif Al Ameri Sana Ziaei in 2013. This device has a lot of functionalities which mainly includes detection and alert, emergency, and renewable power generation [5]. However the extra components which are added to generate electrical energy from the movement of the stick makes the device heavy and difficult to move easily.

# Chapter 3 Design

## 3.1 Software Design

The software part of this system is an image recognition application that uses camera to capture the images in real time and process the captured image on the raspberry pi.

The application captures the live video from the USB web cam and then it loads the pre-trained classifier. And using the classifier it test each frame for the target object.

The general flow diagram of the image detection software is shown as follows.

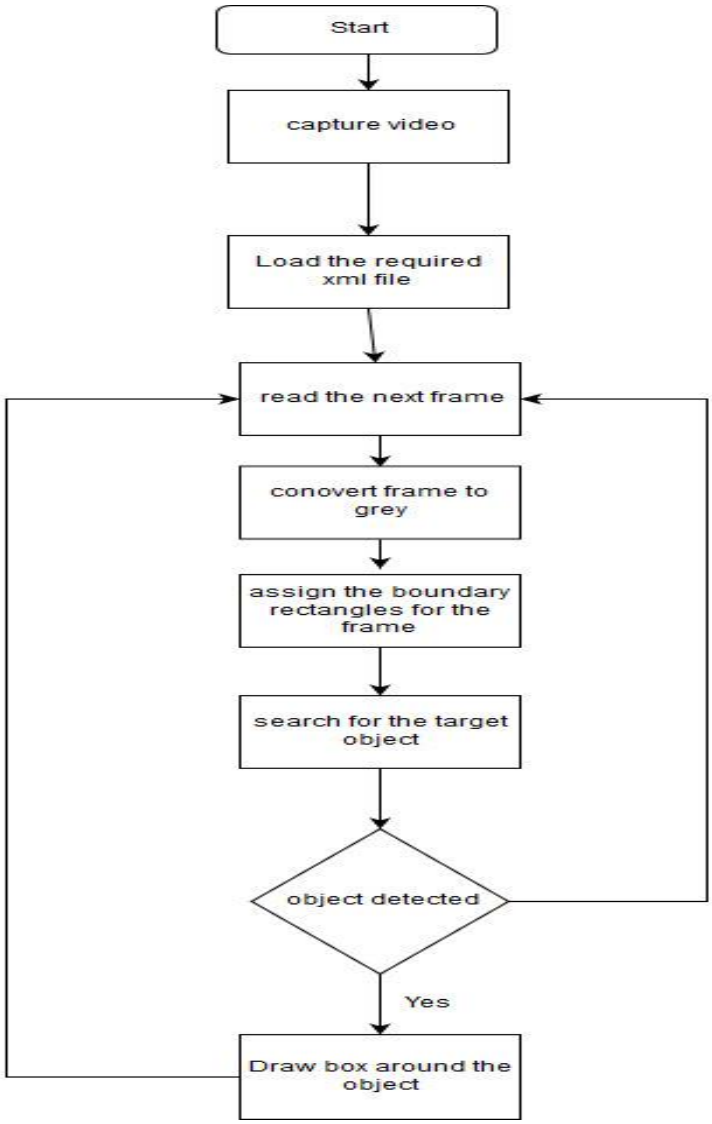


*Figure 1: flow chart of the designed system*

### 3.1.1 Coding platform and selection

We have used python as programming language for the coding of this system.

## Python

Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options. It is simple, so it's easy to learn since it requires a unique syntax that focuses on readability.
While languages such as C and C++ are compile (i.e. turn directly into computer code), it has not use by the Python interpreter. This does mean that Python is slower than the C and C++ languages. However, Python — and most of its libraries — are cross-platform, which means the code written on a Windows machine requires little or no change to run on a Mac OS or Linux [6]

## 3.2 Hardware Design

The hardware part of this system is composed of sensors and the raspberry microcontroller. The main components of this system are proximity sensors, moisture sensor, light sensor, USB camera and buzzer.

### 3.2.1 System Block Diagram

The overall system block diagram is shown as follows.

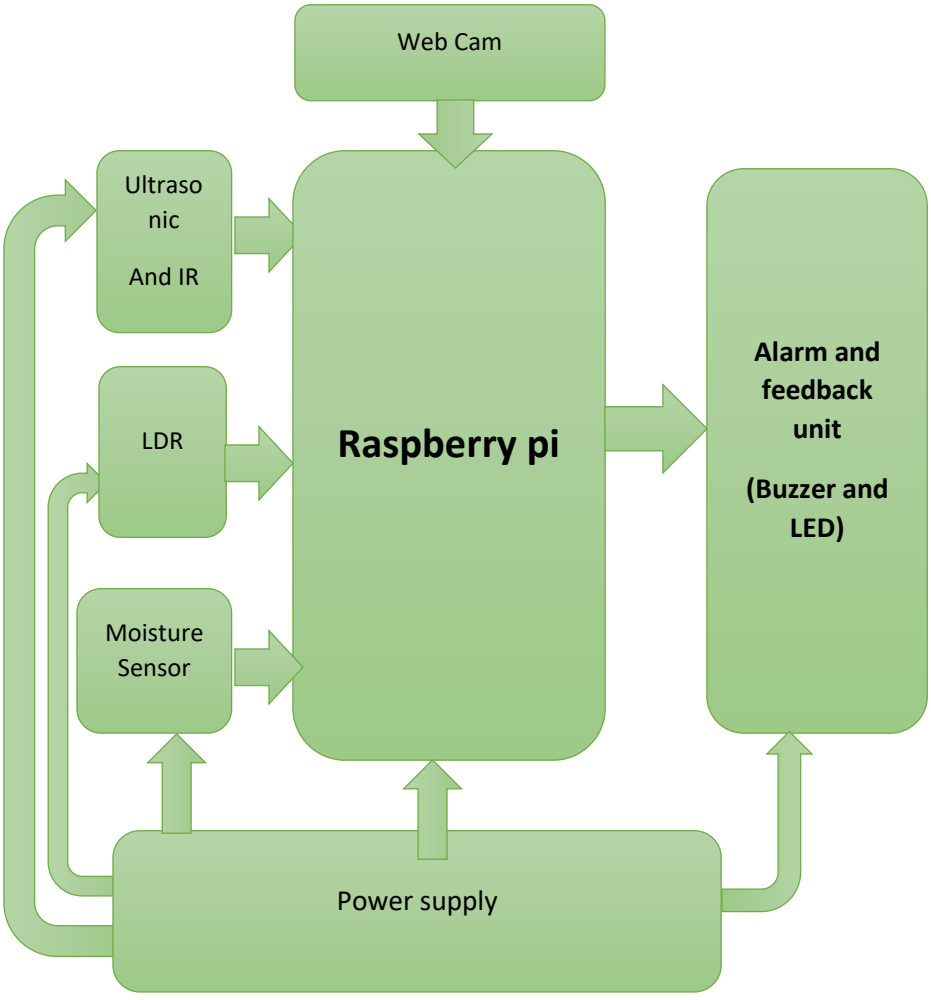


*Figure 2: block diagram of the system's overall design*

### 3.2.2 Components Selection

Generally the device is designed to mimic human obstacle detection and vision capabilities so that it can closely match the eye based on natural environmental perception. Therefore the device requires proximity sensors to detect big obstacles, small obstacle detectors, light intensity detectors, temperature sensors to detect fire areas, moisture sensors to sense the existence of muddy and watery places and USB web camera to detect objects using image recognition. And the microcontroller which is the brain of this device and makes decisions and inform the user by gathering different kinds of information from the environment through the aforementioned sensors and processing it.

## I. Microcontroller

The raspberry pi and the Arduino Uno was both candidates as the microcontrollers for this device during the selection of components. They are both open source and come with a wealth of online documentation and help. Even though the Arduino microcontroller is cheaper and easier to work with, it is doesn't have as powerful computing capability as the Raspberry pi. Since this device uses image processing for camera based object detection. The raspberry pi is by far the better and reasonable option to use as the microcontroller for this project.

*Figure 3 Raspberry Pi 3-MODB-1GB*

https://www.amazon.in/Raspberry-Pi-Model-RASP-PI-3-Motherboard/dp/B01CD5VC9

## II. Proximity Sensors

Proximity sensors are noncontact object detectors. Therefore they are the best candidates for the implementation of big obstacles detection. There are different kinds of proximity sensors out there in the engineering world. To mention few, Inductive sensors detect metallic objects, capacitive sensors detect all kinds of materials and there are ultrasonic sensors that detect all sound reflective materials. Because of their good functionality in a wide range of environmental factors, from dark and condition area to bright and light polluted place because their means of working is an ultrasonic sound which is out of the range of human hearing level. They have good range of detection. Due to these advantages on other proximity sensors the ultrasonic sensor is chosen to detect large obstacles.

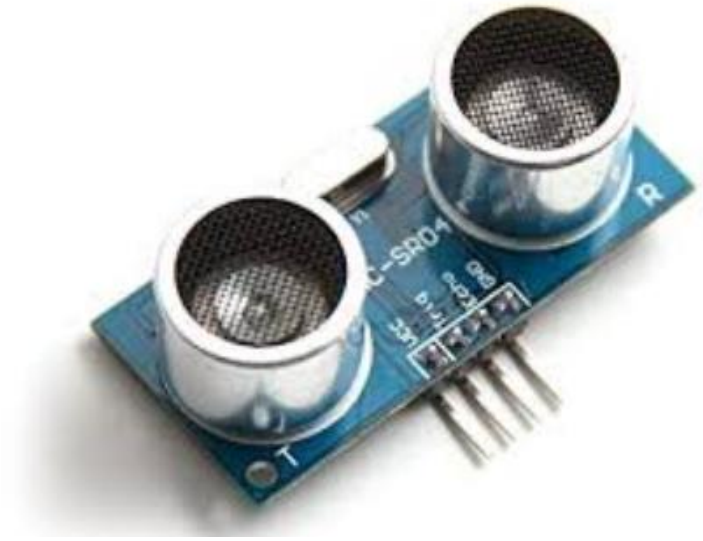


*Figure 4 HC-SR04 Ultrasonic Distance Measuring Sensor*

https://www.eprlabs.com/product/hc-sr04-ultrasonic-distance-measuring-sensor-module/

## III. Light Sensors

Light Sensors are photoelectric devices that convert light energy (photons) whether visible or infra-red light into an electrical signal. To detect the light intensity the Light Dependent Resistor (LDR) is used because of its low cost and availability.

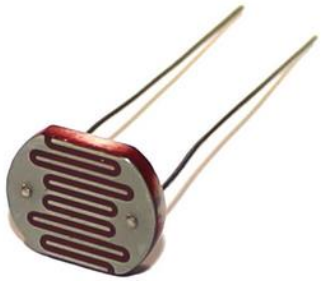

*Figure 5 Light Dependent Resistor (LDR)*

https://www.kitronik.co.uk/blog/how-an-ldr-light-dependent-resistor-works/

## IV. Small Obstacle Detectors

Even though ultrasonic sensors are known for their detection of obstacles using high frequency sound echoes, when it comes to measuring short distances and detecting small obstacles their accuracy is not satisfactory enough. Due to this an infrared sensor is used to detect small obstacles. An Infrared sensor detects an obstacle by emitting infrared light towards the obstacle and detecting the reflected signal.

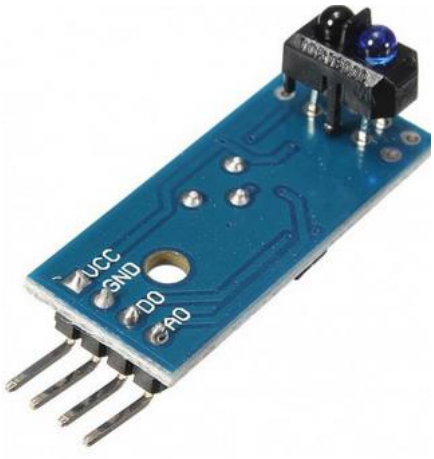


*Figure 6: IR REFLECTIVE SENSOR - TCRT5000*

https://leeselectronic.com/en/product/16041.html

## V. Moisture Sensor

Water detection is one necessary functionality that is implemented in this device. When the user walks around the device has to detect the existence of water so that he/she can avoid it. There are two main moisture sensors based on their working principle; capacitive and resistive. The resistive sensors has corrosive behavior after they have been exposed to water while current is flowing through them. Therefore we have chosen capacitive sensor for our device.

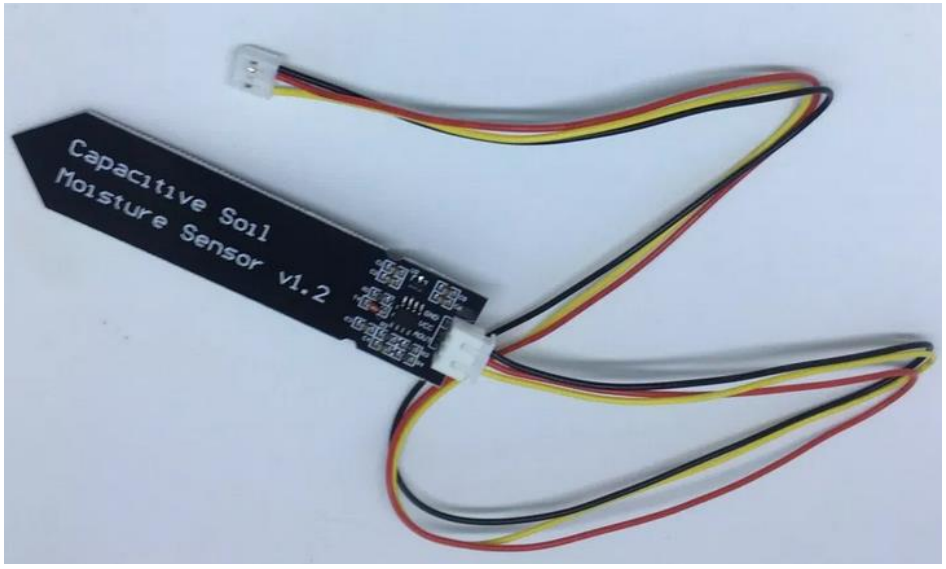


*Figure 7: Capacitive soil moisture sensor*

https://www.switchdoc.com/2018/11/tutorial-capacitive-moisture-sensor-grove/

### 3.2.3 Temprature Sensors

Temprature measurement is necessary to make the device detect fire areas. Therefore we have included a temprtature measurement system in our device.

The first candidate for temprature senosr was the LM35 temprature sensor becasuse it has directly propertional out put voltage to the degree centigrade scale. How ever since the out of this device is analog quantity it would add another module i.e. the ADC, because the raspberry pi doesn't read analog inpute. which adds to the power consumption and price of the device. For this reason we have used a thermister to measure the temprature of the environment.

#### Thermister

The Thermistor is a special type of variable resistive element that changes its physical resistance when exposed to changes in temperature. It is a two terminal resistive transducer which changes its resistive value with changes in surrounding ambient temperature, hence the name thermal-resistor, or simply "thermistor".

Thermistors are inexpensive, easily-obtainable temperature sensors constructed using semiconductor metal oxides, and are available with either a negative temperature coefficient, (NTC) of resistance or a positive temperature coefficient (PTC) of resistance. The difference being that NTC thermistors reduce their resistance as the temperature increases, while PTC thermistors increase their resistance as the temperature increases [7].


*Figure 8: Thermister*

https://www.electronics-tutorials.ws/io/thermistors.html

### 3.2.4 Alarm sytem

For the device to interact with user an alarm system mainly build from a buzzer is used.

## KY-006 Passive Piezo-Buzzer module

it a an electrical buzzer which can produce a range of sound tones depending on the input frequency. The KY-006 Buzzer Module consists of a passive piezoelectric buzzer, it can generate tones between 1.5 to 2.5 kHz by switching it on and off at different frequencies either using delays or PWM [8]


*Figure 9:KY-006 Passiv Piezo-Buzzer module*

https://circuitsfun.com/products/ky-006-passive-buzzer-module

### 3.2.5 System Description

The designed system is a smart device that detects obstacles, environmental conditions and identifiable objects using different sensors and USB camera on raspberry pi microcontroller. It will be the replacement and enhancement the ordinary blind stick. It has three ultrasonic sensors mounted on the three directions (front, right and left), one Infrared sensor at the bottom of the stick, one LDR, one temperature sensor, and one USB cam. The main feature of the smart Kane are:
It detects large obstacles without any contact using ultrasonic sensors
It detects small obstacles using Infrared sensor
It detects the light intensity using LDR
It measures the environmental temperature for possible fire hazards using temperature sensors
It uses camera to detect stairs and cars

The step by step description of the system is presented as follows.
First the front ultrasonic sensor measures the front distance to detect if there is any obstacle in front of the user. If there is any obstacle at 2m distance the right and left sensors measure the distance on both directions to decide which way the user should turn to and send this information to the user. The infrared sensor at the bottom of the stick detects if there is any obstacle in front of the user constantly and sends warning to the user. The LDR and the temperature sensor measures the light intensity and the temperature sensor respectively

### 3.2.6 Implemented System flow chart

The flowchart of the implemented system is shown as follows.

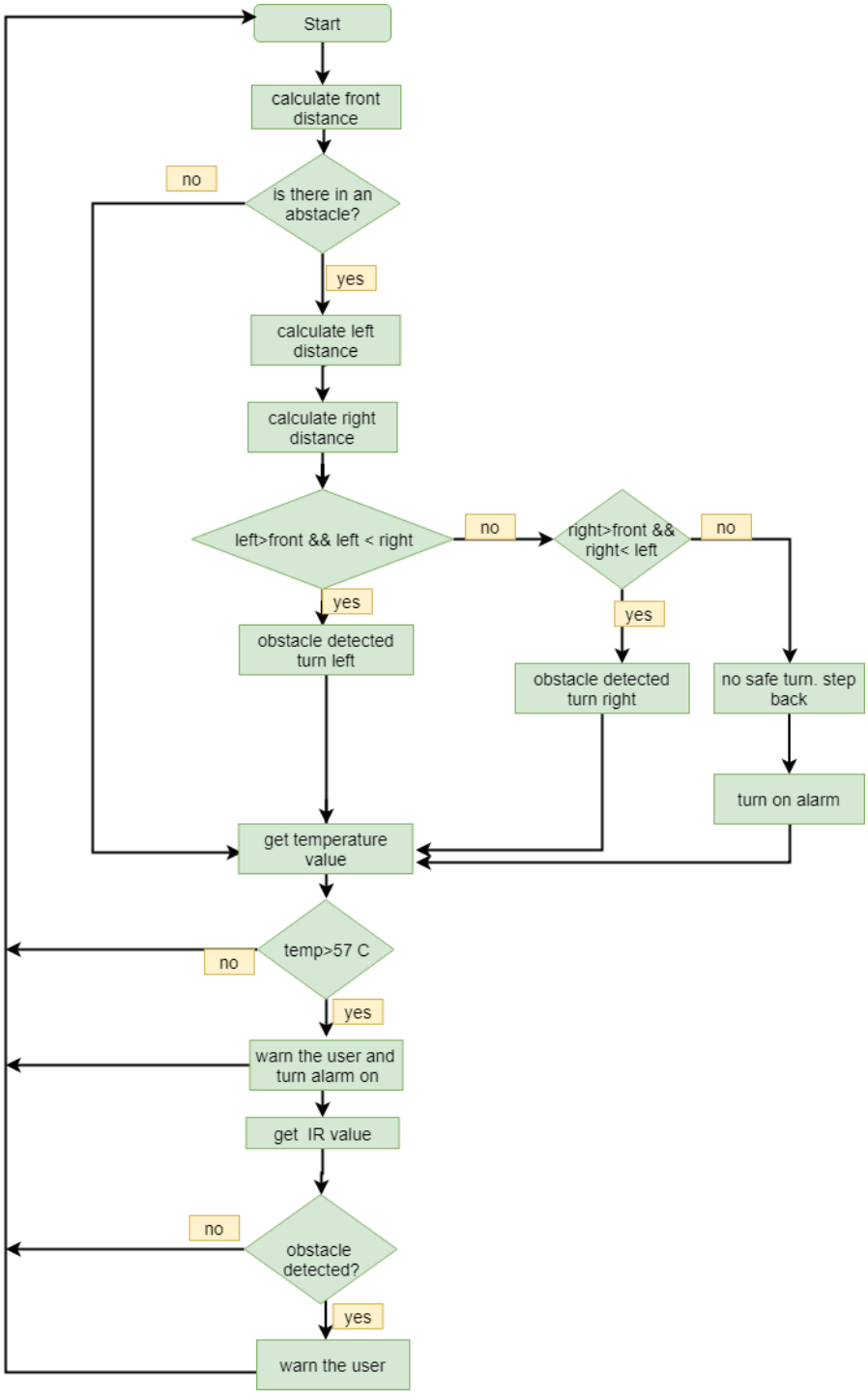


*Figure 10: flow chart of the whole system*

# Chapter 4 Implementation

This device is designed to have to main components i.e. the software and hardware components. The hardware component which has the raspberry pi microcontroller as its central brain contains different kinds of sensors to detect the state of the environment and send the gathered information to the microcontrollers. The microcontroller makes the desired decision based on the sensor data and directs the user accordingly.

The software component is a camera based image recognition system. This system uses OpenCV library and trained Haar cascade classifier to detect stairs and cars.

## 4.1 Hardware

The hardware system mainly contains sensors, alarm system and response systems and the Raspberry pi microcontroller.

### 4.1.1 Obstacle detection mechanism

The main component of this device is the obstacle detection block. Therefore to detect the presence of these obstacles the method we have to use is measuring their distance from the user using ultrasonic sensors.

#### HC-SR04 Ultrasonic Sensor

An ultrasonic sensor is a type of active sensor which uses sonar to measure the distance of large obstacles. The HC-SR04 Ultrasonic Sensors is one kind of the ultrasonic sensors which offers an excellent noncontact range detection with high accuracy and stable reading in an easy to use package. It emits an ultrasound wave at 40,000 Hz which travels through the air and if there is an object or obstacle on its path it will bounce back to the module. Given the travel time and the speed of the sound it is possible to calculate the range of the distant obstacle without any physical contact with object.


*Figure 11: the working of ultrasonic sensor*

https://components101.com/ultrasonic-sensor-working-pinout-datasheet

The distance is calculated as follows

Distance = (sound speed * time)/2

Distance = 34300cm/c*time/2

Distance = 17150*time

The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo.

Generally the specification of the ultrasonic sensor module is given as follows

## HC-SR04 Specifications and pin descriptions

- Working Voltage: DC 5V
- Working Current: 15mA
- Working Frequency: 40Hz
- Max Range: 4m
- Min Range: 2cm
- Measuring Angle: 15 degree
- Trigger Input Signal: 10µS TTL pulse
- Echo Output Signal Input TTL lever signal and the range in proportion
- Dimension 45 * 20 * 15mm [9]

| No. | Pin Name | Pin Description |
|---|---|---|
| 1 | VCC | The power supply pin of the sensor that mainly operates at 5V DC. |
| 2 | Trig Pin | It plays a vital role to initialize measurement for sending ultrasonic waves. It should be kept high for 10us for triggering the measurement. |
| 3 | Echo Pin | This pin remains high for short period based on the time taken by the ultrasonic waves to bounce back to the receiving end. |
| 4 | Ground | This pin is connected to ground. |

*Figure 12: Specification and pin description of HC-SR04 Specifications*

https://www.theengineeringprojects.com/2018/10/introduction-to-hc-sr04-ultrasonic-sensor.html

In order to generate the ultrasound wave we need to set the Trig on a High State for 10 µs. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received by the Echo pin. The echo pin of the ultrasonic sensor stays at high state until it receives the reflected wave. The state of the pin is measured with time by connecting it to a GPIO pin on the raspberry pi.

The Echo Pin of the ultrasonic sensor has an output of 5v. However the GPIO pins of the raspberry pi can only tolerate 3.3v. Therefore a simple voltage divider is used to connect it to the GPIO pins

The time pulse and finally the distance measured in centimeter is measured as follows.

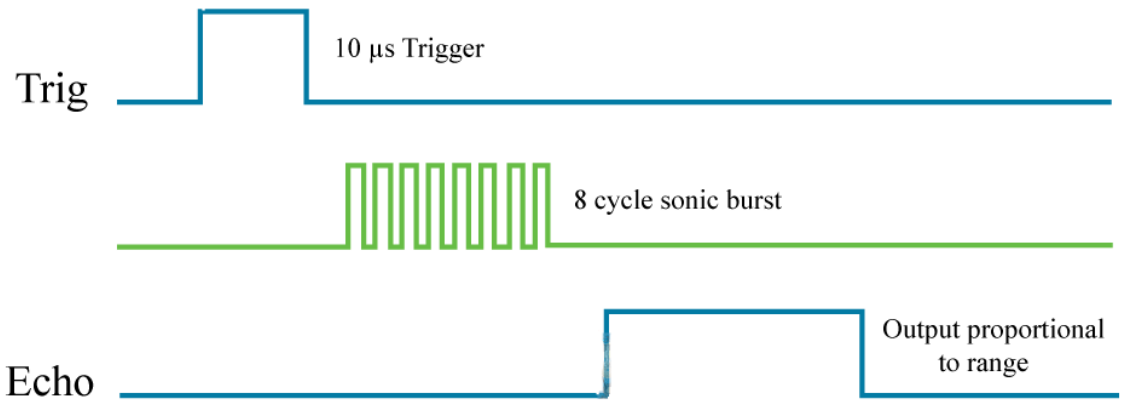


*Figure 13: distance measurement mechanism in ultrasonic sensors*

https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

The distance calculated using this method is then used to decide how far the object is and inform the user accordingly. The designed device uses three ultrasonic sensors which are mounted on left side, front and right side. These three ultrasonic sensors generates ultrasonic wave constantly to detect obstacles on either side. E.g. if an obstacles detected in front of the user then the left distance and the right distance are compared to decide which direction the user should turn to.

For visually impaired person, detecting large obstacles is not enough to ensure safe travel. The device also need to be capable of detecting small obstacles. To achieve this goal we have used an active infrared sensor which can detect small objects with excellent accuracy.

### Infrared Sensor

The basic concept of an infrared sensor is to transmit an infrared signal which bounces from the surface of an object and the signal is received at the infrared receiver. The working principle is exactly the same as the ultrasonic sensor except here we have infrared signal instead of ultrasonic signal.

The infrared sensor is mounted at the bottom of the device so that it can detect small objects that are near the surface.

### 4.1.2 Light detection mechanism

When the user walks at night it is necessary to notify others that there is someone walking to avoid collision. It is also important to notify the user that it is getting darker. In order to achieve this we have used LDR. LDR is light dependent resistor whose resistance values based on the light intensity it receives

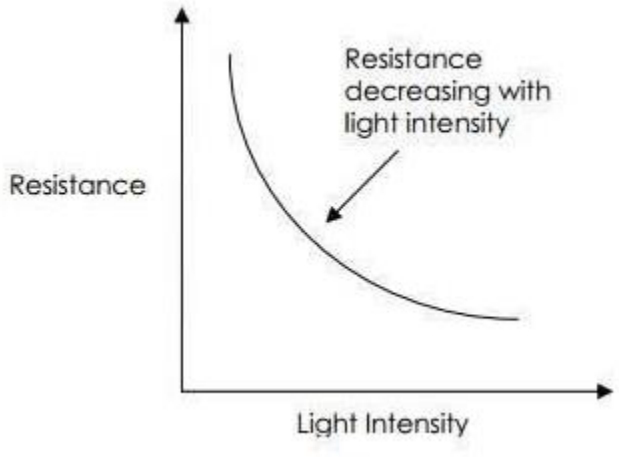


*Figure 14: Light Intensity Vs Resistance characteristics curve of an LDR*

https://www.kitronik.co.uk/blog/how-an-ldr-light-dependent-resistor-works/

The equation of LDR's characteristic curve.

$$Rf = AE^{-\alpha}$$

Where E is luminous energy in Lux, $Rf$ is the resistance; A and α are constants given by the material.
The output of the LDR is an analog resistance which needs an ADC to measure it because the raspberry pi doesn't have an Analog GPIO pin. To measure the resistance level of the LDR without an ADC we need to use another way which is connecting it with a capacitor in series and measure the resistance level from the charging time of the RC circuit.
Charging time = RC
R = charging time/C

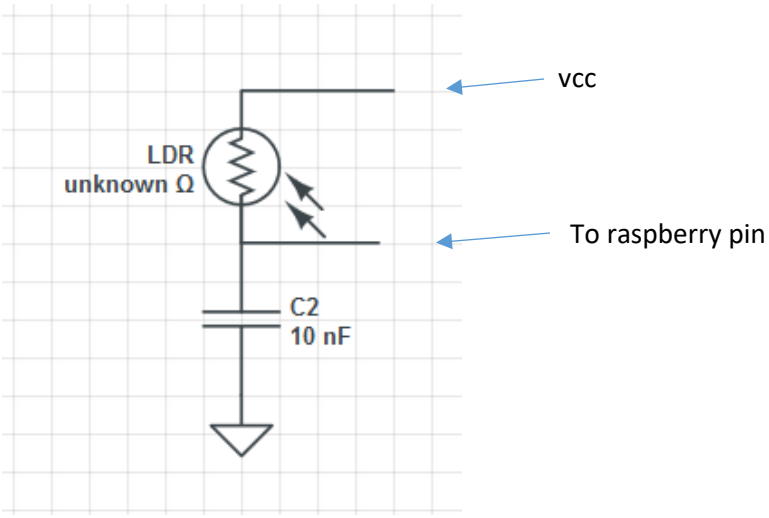


*Figure 15: circuit diagram of the LDR*

### 4.1.3 Measuring moisture and water

To measure the existence of water we have used a capacitive moisture sensor. A capacitive moisture sensor works by measuring the changes in capacitance caused by the changes in the dielectric. Basically it measures the dielectric that is formed by the soil and the water which the most important factor that affects the dielectric.

The selected moisture sensor has the following description

#### Specification

- Operating Voltage: 3.3 ~ 5.5 VDC
- Output Voltage: 0 ~ 3.0VDC
- Operating Current: 5mA
- Interface: PH2.0-3P
- Dimensions: 3.86 x 0.905 inches (L x W)
- Weight: 15g [10]

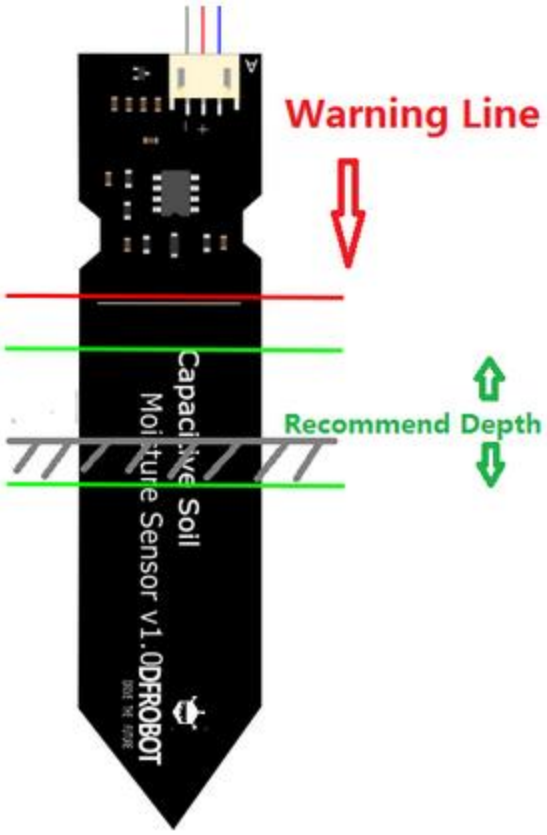


*Figure 16: calibration diagram of moisture sensor*

https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193

The output of the moisture sensor is an analog quantity which means an Analog to digital convertor is needed to read the result. However we can measure the capacitance of the sensor in another way using the charging time method.

## Assembled system

The assembled sytem has the following main components

- Sensors
- Power bank
- Circuit board
- Raspberry pi

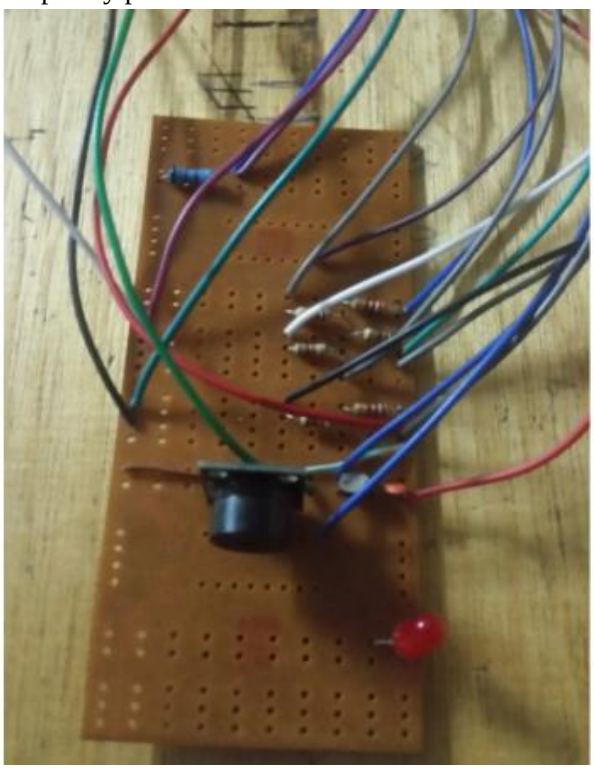

*Figure 17: Circuit Board*


*Figure 18:Assembled Device*

## 4.2 Software

The software part of this device is a camera based image recognition system. It is used to detect cars and stairs using a webcam connected to raspberry. It receives live image from the camera and pass it through trained Haar cascade classifiers to detect weather the image contains car or stair. It is built using openCV library.

**Image Recognition Using HAAR Cascade**

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video. It is machine Learning based approach where cascade function is trained from a lot of positive and negative images which is then used to detect objects in other images based on the feature extracted in the training process. [11]

The algorithm has four stages:

i. Haar Feature Selection

A Haar features is a digital image feature which considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. To make the computation of the sums more fast and effective a method called an Integral Image is used.

ii. Integral Images

Integral Images is an algorithm for quickly and efficiently generating the sum of values in a rectangular subset of a grid. Rectangle features can be calculated rapidly using an immediate representation for the image which is called integral image. The integral image at location x, y contains the sum of pixels above and to the left of x, y including the point x, y. [12]

$$i, i(x,y) = \sum_{x' \le x, y' \le y} i(x', y')$$

Let's take the rectangle below with sub rectangles A, B, C, and D and let the point 1 hold the value of the sum of the pixels in rectangle A, the point 2 hold the value of the sum of the pixels in rectangles A+B, point 3 A+C and point 4 A+B+C+D.

The sum of pixels at sub rectangle D can be calculated as follows

4 = A+B+C+D
= 2+3-1+D
D =4+1-(2+3)

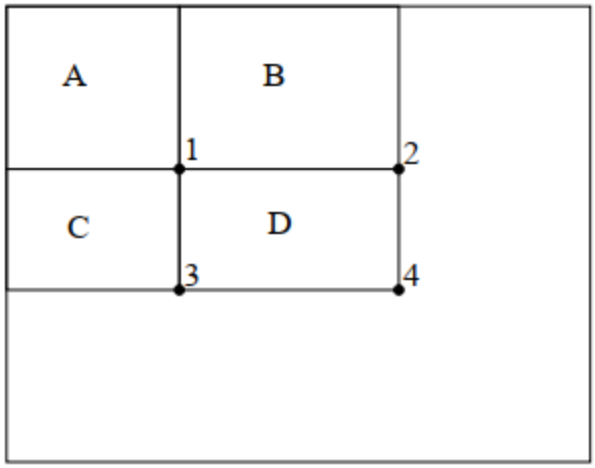


*Figure 19: working of Haar classifier*

Using the integral image any rectangular sum can be calculated using four array references. The Haar feature is calculated by finding the difference between the sums of adjacent regions.

iii. Adaboost

However Among all the Haar features calculated most of them are irrelevant. The selection of the best features out of calculated Haar features is done using an algorithm called Adaboost. It is also used to train the classifier based on the selected best features. This algorithm constructs a strong classifier out of cascaded combination of weak classifiers. The process of detection is explained as follows.

During the detection phase a window of the target size is moved over the image and for each subsection of the image Haar features are calculated. The difference is then compared with the learned threshold. Even though each Haar feature is weak classifier the classification is based on numerical data which helps collectively.

iv. Cascade Classifier

The cascade classifier consists of a collection of stages, where each stage is an ensemble of weak learners. The weak learners are simple classifiers called decision stumps. Each stage is trained using a technique called boosting. Boosting provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. Positive indicates that an object was found and negative indicates no objects were found. If the label is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as positive.
[11]
The stages are designed to reject negative samples as fast as possible. The assumption is that the vast majority of windows do not contain the object of interest. Conversely, true positives are rare and worth taking the time to verify.
Three types of results are produced by cascade classifier.

- A true positive occurs when a positive sample is correctly classified.
- A false positive occurs when a negative sample is mistakenly classified as positive.
- A false negative occurs when a positive sample is mistakenly classified as negative.

To work well, each stage in the cascade must have a low false negative rate. If a stage incorrectly labels an object as negative, the classification stops, and you cannot correct the mistake. However, each stage can have a high false positive rate. Even if the detector incorrectly labels an object as positive, you can correct the mistake in subsequent stages. Adding more stages reduces the overall false positive rate, but it also reduces the overall true positive rate.

Cascade classifier training requires a set of positive samples and a set of negative images. The positive samples are provided with region of interest labeled.
Open CV libraries are used to train and test the cascade classifier. Because Open CV provides built in applications and libraries to train and test the Haar cascade classifiers. The main training applications from open CV are: opencv_createsamples, opencv_annotation, opencv_traincascade and opencv_visualisation. In this system two cascade classifier are trained i.e. Car detector and stair detector.
The following steps are followed in training the detectors.

**Steps of Training a Cascade Classifier**

**I. Collecting data**

Data is the main source of artificial intelligence system. An object detection system needs to be trained using a lot of data before it can detect the object it is supposed to.

Therefore the first step in designing and building an image recognition system is training it on huge amount of data. There are two types of sample data that are used to train Haar cascade classifier; positive and negative. Negative samples correspond to non-object samples and positive samples correspond to objects that are needed to be detected.

During training the cascade classifier for the stair detector 1000 positive images and 2030 negative images were used as inputs.

**II. Processing data**

The collected data should be edited and processed in such a way that it can be used to for the classifier training which includes cropping, renaming, and resizing. Some of the image processing and preparation were accomplished automatically using python scripts.

**III. Cascade Training**

The next step is the actual training of the boosted cascade of weak classifiers, based on the positive and negative dataset that was prepared beforehand. Command line application opencv_traincascade is used to train the system with prepared samples with following command line arguments.

- -data <cascade_dir_name> : Where the trained classifier should be stored. A folder created manually beforehand.
- -vec <vec_file_name> : vec-file with positive samples (created by opencv_createsamples utility).
- -bg <background_file_name> : Background description file containing the negative sample images.
- -numPos <number_of_positive_samples> : Number of positive samples used in training for every classifier stage.
- -numNeg <number_of_negative_samples> : Number of negative samples used in training for every classifier stage.
- -numStages <number_of_stages> : Number of cascade stages to be trained.
- -acceptanceRatioBreakValue <break_value> : This argument is used to determine how precise our model should keep learning and when to stop. A good guideline is to train not further than 10e-5, to ensure the model does not overtrain on your training data. By default this value is set to -1 to disable this feature [13].

# Chapter 5 Results

## 5.1 Sensor results

The results of the sensors and cmponent devices is shown as follows

a. Mositure sensor

Since the moisture sensor is a cacpacitve sensor it is measured indirectley using an RC ciruit.

**Capacitance in air**

Charging time = 165 microsecond

Used resistance =12K ohm

T =RC

C = T/R

C= $165*10^{-9}/12*10^{3}$ F

C =13.75nf

**Capacitance in water**

Charging time = 175 microsecond

Used resistance =12K ohm

T =RC

C = T/R

C= $165*10^{-9}/12*10^{3}$ F

C =14.58nf

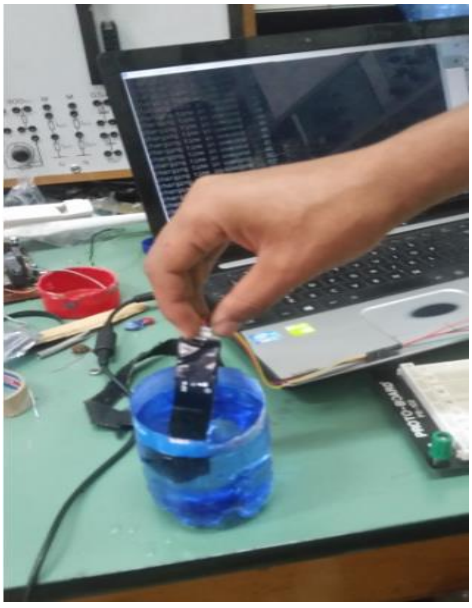

*Figure 20: capacitance measurement of moisture sensor*

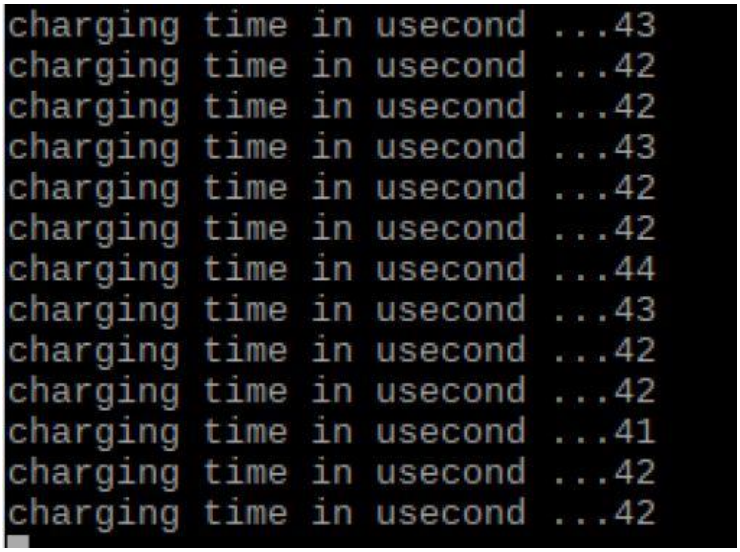


*Figure 21:Charging time of th moisture sensor on air*

b. Ultrasonic sensor

Accuracy of ultrasonic sensor

| No | Actual Distance in cm | Measure Distance In cm | Error % |
|---|---|---|---|
| 1 | 4 | 3.98 | 0.24875 |
| 2 | 6 | 6.1 | 0.169444 |
| 3 | 8 | 7.81 | 0.122031 |
| 4 | 10 | 10.1 | 0.101 |
| 5 | 12 | 12.2 | 0.084722 |
| 6 | 14 | 14.11 | 0.07199 |
| 7 | 16 | 15.8 | 0.061719 |
| 8 | 18 | 18.2 | 0.056173 |
| 9 | 20 | 20.1 | 0.05025 |
| 10 | 22 | 22.11 | 0.045682 |
| 11 | 24 | 23.6 | 0.040972 |
| 12 | 26 | 26.3 | 0.038905 |
| 13 | 28 | 28.2 | 0.035969 |
| 14 | 30 | 29.8 | 0.033111 |

*Figure 22: Accuracy of Ultrasnic senso*

**Image recogniton results**

The car detection and stair detection are tested with test videos and the results are shown below.

car detection system

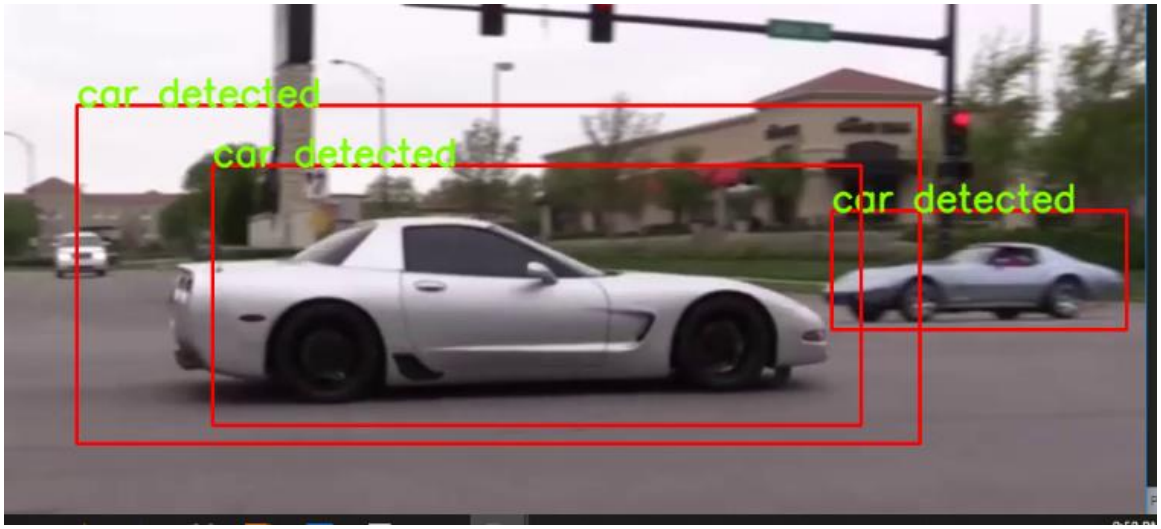


*Figure 23: care detecion system at work*

Stair Detection System


*Figure 24: stair detection mechanism*

# Chapter 6 Conclusion and Recomendation

Generally the device smart Cane for blind people includes two main components i.e. software and hardware components. The software component is an application which detects objects using camrea. It is based on openCv library and written in python. It is designed and developed to detect cars and stairs

The hardware component is mainly made of the sensors and the raspberry pi microcontroller.Using the sensors we have designed and buit a smart kane wich detects obstacles using proximity sensors and senses the condition of the environmenst using moisture sensor,LDR and temprature sensor.

The main thing we have learned in designing this device is the variety of different sensors and electrical components and their area of application. In addition we have learned how to select the right component for a specific application based on the given system requirements and specified features of the component in consideration.

We have learned and implemented the concepts of computer vision to detect cars and stairs. During the trianing phase of the stair detecter system we have observed that it needs a lot of data set to make the classifier as accurate as possible. The depths and edges features of stairs could be detected more accurately using streo camera system. Therefore we recommend that stereo camera based object detection better captures the real nature of objects.

# Chapter 7 References


[1] W. H. O. (WHO), "Global Data on Visual Impairments," World Health Organization (WHO), 2012.

[2] M. F. K. a. S. R. Mithiles Kumar, "Low Cost Smart Stick for Blind and Partially," *International Journal of Advanced Engineering and Management,* vol. Volume 2, no. Issue 3, pp. 65-68, 2017.

[3] N. S. a. C. U. Oluwaseyitan Joshua Durodola, "Blind Assist: Project Report," Howard University, Department of Electrical and Computer Engineering, 2012.

[4] A. I. S. #. A. L. A. Dada Emmanuel Gbenga #1, "Smart Walking Stick for Visually Impaired," University of Maiduguri, Maiduguri, Borno State, Nigeria, October 2017.

[5] H. M. A. A. a. S. Z. Mohammad, "Self-Energized Smart Vision Stick for Visually Impaired People," American University Dubai, Electrical Engineering, Dubai, 2013.

[6] SourceCode and Projects., "Why Python Is Used In Raspberry Pi," SourceCode and Projects., [Online]. Available: https://code-projects.org/why-python-is-used-in-raspberry-pi/. [Accessed 01 07 2019].

[7] AspenCore, "Electronics Turorials," AspenCore, 22 05 2017. [Online]. Available: Thermistors. [Accessed 01 07 2019].

[8] TkkR Lab, "Arduino KY-006 Small passive buzzer module," TkkR Lab, 7 6 2016. [Online]. Available: https://circuitsfun.com/products/ky-006-passive-buzzer-module. [Accessed 2019 6 22].

[9] ElecFreak, Ultrasonic Ranging Module HC - SR04.

[10] Capacitive_Soil_Moisture_Sensor.

[11] W. Berger, "Deep Learning Haar Cascade Explained".

[12] v. jones, "Rapid Object Detection using a Boosted Cascade of Simple".

[13] A. M. &. A. K, OpenCV-Python Tutorials Documentation - Release 1, Nov 05, 2017.

[15] S. Productions, " Python Definition," Python Definition, 15 01 2010. [Online]. Available: https://techterms.com/definition/python&as_qdr=y15. [Accessed 1 7 2019].

# Chapter 8 Appendix

## A. Python code of ultrasonic sensor

```
GPIO.setmode(GPIO.BCM)
Buzzer =15 # pin number assigned to the buzzer signal
IR_pin = 16 # pin number assined to Infrared signal
TRIG_front = 4 # trig pin for the front ultrasonic sensor
ECHO_front = 18 # echo pin for the front ultrasonic sensor
TRIG_left = 20  # trig pin for the left ultrasonic sensor
ECHO_left = 21  # echo pin for the left ultrasonic sensor
TRIG_right = 6  # trig pin for the right ultrasonic sensor
ECHO_right = 13 # echo pin for the left ultrasonic sensor
LED_pin = 17 # LED pin number
LDR_pin = 14 #pin from the LDR

                Set up of the all the sensor pins
def GPIO_setup():
    GPIO.setwarnings(False)
    GPIO.setup(Buzzer,GPIO.OUT)
    GPIO.setup(TRIG_front,GPIO.OUT)
    GPIO.setup(ECHO_front,GPIO.IN)
    GPIO.setup(IR_pin,GPIO.IN)
    GPIO.setup(TRIG_left,GPIO.OUT)
    GPIO.setup(ECHO_left,GPIO.IN)
    GPIO.setup(TRIG_right,GPIO.OUT)
    GPIO.setup(ECHO_right,GPIO.IN)
    GPIO.setup(LED_pin,GPIO.OUT)

def get_front_distance(trig,echo):
    end = 0
    start = 0
    GPIO.output(trig, True)
    time.sleep(0.00001)
    GPIO.output(trig, False)
    while GPIO.input(echo) == False:
        start = time.time()
    while GPIO.input(echo) == True:
        end = time.time()
    sig_time = end-start
```

```
    #distance in cemiter
distance = sig_time *17150 #34300cm/s/2
    return distance



def decide_direction(front,left,right):
    if left>200 & left>right:
        print("turn left")
    else if right>200 & right>left:
        print("turn right")
    else:
        print("there is no safe turn return back")
```

### B. LDR sensor measurement using RC circuit

```
def rc_time (pin_to_circuit):
    GPIO.setup(LED, GPIO.OUT)
    GPIO.output(LED,GPIO.LOW)
    count = 0
    GPIO.setup(pin_to_circuit, GPIO.OUT)

    GPIO.output(pin_to_circuit, GPIO.LOW)
    time.sleep(0.1)
    GPIO.setup(pin_to_circuit, GPIO.IN)
    while (GPIO.input(pin_to_circuit) == GPIO.LOW):
        count += 1
    return count


def check_darkness():
   if rc_time(pin_to_circuit)>3000:
       return True
    else:
        return False
```

### IR sensor measurements

```
import RPi.GPIO as GPIO
import time
#definition and setup
GPIO.setmode(GPIO.BCM)
IR_pin = 16
LED_pin = 21

def setup():
    GPIO.setwarnings(False)
    GPIO.setup(IR_pin,GPIO.IN,GPIO.PUD_UP)
    GPIO.setup(LED_pin,GPIO.OUT)
    GPIO.output(LED_pin, False)
```

```
def destroy():
    GPIO.output(LED_pin, GPIO.LOW) # Set the OutLedPin turn HIGH
    GPIO.cleanup() # Release resource
def small_obstacle():
    if GPIO.input(IR_pin)==GPIO.LOW:
        return True
setup()
try:
    while True:
        if small_obstacle():
            GPIO.output(LED_pin, GPIO.HIGH)
        else:
            GPIO.output(LED_pin, GPIO.LOW)
except KeyboardInterrupt: # When control c is pressed child program destroy() will be executed.
    destroy()
```

### C. Image pre-processing python scripts

#### I. Scaling down

```
path = 'E:/Haar Training Data/positive'
basewidth = 32
directory = os.fsencode(path)
current_directory = os.getcwd()
dest_directory ='E:/Haar Training Data/pos'
def edit_pos_images():
        if not os.path.exists(dest_directory):
                        os.makedirs(dest_directory)
        for image in os.listdir(directory):
                file_name = os.fsdecode(image)
                current_image_path=path+'/'+str(file_name)
                dest_image_path = str(dest_directory)+'/'+file_name
                img = Image.open(current_image_path)
                wpercent = (basewidth/float(img.size[0]))
                hsize = int((float(img.size[1])*float(wpercent)))
                img = img.resize((basewidth,hsize), Image.ANTIALIAS)
                gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
                gray.save(dest_image_path)
                # cv2.imwrite(dest_image_path,resized_img)
                print('writing the file'+current_image_path+'to'+dest_image_path)
        print('resizing completed succesfully')
edit_pos_images()
```

### II. Resizing an image

```
import cv2
import numpy as np
import os
import crope

path = 'E:/python_opencv/p'
directory = os.fsencode(path)
current_directory = os.getcwd()
dest_directory =os.path.join(current_directory,r'cropped')


def edit_pos_images():
        i=0
        if not os.path.exists(dest_directory):
                os.makedirs(dest_directory)
        for image in os.listdir(directory):
                        file_name = os.fsdecode(image)
                        current_image_path=path+'/'+str(file_name)
                        img =cv2.imread(current_image_path)
                        rows,cols,channel = img.shape
                        stair = img[300:rows-100,750:cols-850]
                        #resized_img = cv2.resize(img,(100,100))
                        dest_image_path = str(dest_directory)+'/'+file_name
                        cv2.imwrite(dest_image_path,stair)
                        print('writing file.....'+str(i))
                        i+=1
        print('completed succesfully')
def edit_neg_images():
        path_neg = 'E:/python_opencv/n'
        dir_n =os.fsencode(path_neg)
        dest= os.path.join(current_directory,r'negatives2')
        if not os.path.exists(dest):
                os.makedirs(dest)
        img_num = 0
        for image in os.listdir(dir_n):
                file_name =os.fsdecode(image)
                current_image_path= path_neg+'/'+str(file_name)
                img = cv2.imread(current_image_path)
                dest_image_path =str(dest)+'/'+'neg'+str(img_num)+'.jpg'
```

```
            cv2.imwrite(dest_image_path,img)
            img_num+=1
            print(dest_image_path)
            print('writting the file'+current_image_path+'to'+dest_image_path+'/n')
print('resizing and moving completed/n')
print('total number of images = '+str(img_num)
```

### D. Car and stair detection system using webcam

```
import cv2
import numpy as np

cap = cv2.VideoCapture('E:/Project/test videos/cars2.mp4')
car_casecade = cv2.CascadeClassifier('data/sideview_cascade_classifier.xml')
car_cascade2 =cv2.CascadeClassifier('cars.xml')
stair_casecade = cv2.CascadeClassifier('E:/python_opencv/results/training 4/cascade.xml')
def change_res(width,height):
    cap.set(3,width)
    cap.set(4,height)

change_res(640,480)
font = cv2.FONT_HERSHEY_SIMPLEX
while True:
        ret,frame = cap.read()
        gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        cars = car_casecade.detectMultiScale(gray,1.3,5)
        cars2 = car_cascade2.detectMultiScale(gray,1.3,5)
        stairs = stair_casecade.detectMultiScale(gray,1.5,3)
        for (x,y,w,h) in cars:
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)
                cv2.putText(frame,'car detected',(x,y),font,1,(0,255,130),2,cv2.LINE_AA)
        for (x,y,w,h) in cars2:
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)
                cv2.putText(frame,'car detected',(x,y),font,1,(0,255,130),2,cv2.LINE_AA)
        for (x,y,w,h) in stairs:
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)
                cv2.putText(frame,'stair',(x,y),font,1,(0,255,130),2,cv2.LINE_AA)
                cv2.imshow('cap',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
                break
cv2.destroyAllWindows()
```

### E. Car and stair detection system using piCam

```
import cv2
import time
import pygame
import RPi.GPIO as GPIO
import time

class CameraInst():
    # Constructor...
    def __init__(self):
        fps       = 20.0            # Frames per second...
        resolution = (640, 480)       # Frame size/resolution...
        w = 640
        h = 480

        self.cap = cv2.VideoCapture(0)  # Capture Video...
        print("Camera warming up ...")
        time.sleep(1)

        # Define the codec and create VideoWriter object
        fourcc = cv2.VideoWriter_fourcc(*"H264")    # You also can use (*'XVID')
        self.out = cv2.VideoWriter('output.avi',fourcc, fps, (w, h))

    def captureVideo(self):
        # Capture
        self.ret, self.frame = self.cap.read()
        # Image manipulations come here...
        self.gray = cv2.cvtColor(self.frame, cv2.COLOR_BGR2GRAY)
        #cv2.imshow('frame',self.gray)
        return self.frame

    def saveVideo(self):
        # Write the frame
        self.out.write(self.frame)

    def __del__(self):
        self.cap.release()
        cv2.destroyAllWindows()
        print("Camera disabled and all output windows closed...")

def detect():
    font = cv2.FONT_HERSHEY_SIMPLEX
    cam1 = CameraInst()
```

```
    stair =cv2.CascadeClassifier("/home/pi/resources/stair.xml")
    car = cv2.CascadeClassifier("/home/pi/resources/sideview_cascade_classifier.xml")
    face_casecade =
cv2.CascadeClassifier('/home/pi/resources/haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier('/home/pi/resources/haarcascade_eye.xml')
    #cap = cv2.VideoCapture("/home/pi/resources/cars2.mp4")
    while(True):
        frame = cam1.captureVideo()    # Live stream of video on screen...
        #ret,frame = cap.read()
        cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        stairs = stair.detectMultiScale(gray,1.3,5)
        cars = car.detectMultiScale(gray,1.3,5)
        faces = face_casecade.detectMultiScale(gray,1.3,5)
        for (x,y,w,h) in cars:
            cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)
            cv2.putText(frame,'car_detected',(x,y),font,1,(0,255,130),2,cv2.LINE_AA)
        for (x,y,w,h) in stairs:
            cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)
            cv2.putText(frame,'stair_detected',(x,y),font,1,(0,255,130),2,cv2.LINE_AA)

        cv2.imshow('cap',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
    cleanUp()
```

### F. Interfacing with the sensors

```
import pygame
pygame.mixer.init(frequency=19000, size=-16, channels=2, buffer=4096)
pygame.init()
import RPi.GPIO as GPIO
import time
#pin definition .....................................

GPIO.setmode(GPIO.BCM)
Buzzer =15 # pin number assigned to the buzzer signal
IR_pin = 16 # pin number assined to Infrared signal
TRIG_front = 2 # trig pin for the front ultrasonic sensor
ECHO_front = 3 # echo pin for the front ultrasonic sensor
TRIG_left = 14  # trig pin for the left ultrasonic sensor
ECHO_left = 15  # echo pin for the left ultrasonic sensor
TRIG_right = 17  # trig pin for the right ultrasonic sensor
ECHO_right = 27 # echo pin for the l eft ultrasonic sensor
LED_pin = 21 # LED pin number
LDR_pin = 20 #pin from the LDR
moisture_pin =23
#.....................................................
def LED_on():
   GPIO.output(LED_pin, True)
def LED_off():
   GPIO.output(LED_pin, False)
def play_sound(path):
   pygame.mixer.music.load(path)
   pygame.mixer.music.play(0)
   clock = pygame.time.Clock()
   clock.tick(10)
   while pygame.mixer.music.get_busy():
      pygame.event.poll()
      clock.tick(10)
      time.sleep(0.5)
def GPIO_setup():
   GPIO.setwarnings(False)
   GPIO.setup(Buzzer,GPIO.OUT)
   GPIO.setup(TRIG_front,GPIO.OUT)
   GPIO.setup(ECHO_front,GPIO.IN)
   GPIO.setup(IR_pin,GPIO.IN)
   GPIO.setup(TRIG_left,GPIO.OUT)
   GPIO.setup(ECHO_left,GPIO.IN)
   GPIO.setup(TRIG_right,GPIO.OUT)
   GPIO.setup(ECHO_right,GPIO.IN)
```

```
    GPIO.setup(LED_pin,GPIO.OUT)
#claculate the charging time of the capacitor
def rc_time_moisture (pin_to_circuit):
    count = 0
    GPIO.setup(pin_to_circuit, GPIO.OUT)
    GPIO.output(pin_to_circuit, GPIO.LOW)
    time.sleep(0.1)
    GPIO.setup(pin_to_circuit, GPIO.IN)
    while (GPIO.input(pin_to_circuit) == GPIO.LOW):
        time.sleep(0.000001)
        count += 1
    return count
def check_moisture():
    time.sleep(0.5)
    t = rc_time(rc_pin)
    if t>45:
        return True
    else:
        return False
def rc_time (pin_to_circuit):
    GPIO.setup(LED_pin, GPIO.OUT)
    GPIO.output(LED_pin,GPIO.LOW)
    count = 0
    GPIO.setup(pin_to_circuit, GPIO.OUT)

    GPIO.output(pin_to_circuit, GPIO.LOW)
    time.sleep(0.1)
    GPIO.setup(pin_to_circuit, GPIO.IN)
    while (GPIO.input(pin_to_circuit) == GPIO.LOW):
        count += 1
    return count
def check_darkness():
    if rc_time(LDR_pin)>3000:
        ptint(rc_time(LDR_pin))
        return True
    else:
        return False
def get_distance(trig,echo):
    pulse_start =0
    pulse_end = 0
    GPIO.output(trig, False)
    time.sleep(0.5)
    GPIO.output(trig, True)
    time.sleep(0.00001)
    GPIO.output(trig, False)
    while GPIO.input(echo)==0:
```

```
        pulse_start = time.time()
    while GPIO.input(echo)==1:
        pulse_end = time.time()
    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)
    if distance > 2 and distance < 400:
        return distance - 0.5
    else:
        return -1

def obstacle_detected(distance):
    if distance<=200:
        return True
    else:
        return False
def buzzer_beep():
    GPIO.output(15, True)
    time.sleep(0.001)
    GPIO.output(15, False)
def decide_direction(front,left,right):
    if (front>0 and left>0) and right>0:
        if left>200 and left>right:
            play_sound("/home/pi/voices/Turn Left.wav")
        elif right>200 and right>left:
            #print("turn right")
            play_sound("/home/pi/voices/Turn Right.wav")
        else:
            #print("there is no safe turn return back")
            play_sound("/home/pi/voices/step back.wav")
def small_obstacle():
    return GPIO.input(IR_pin)
def destroy():
    GPIO.cleanup() # Release resource
def loop():
    while True:
        time.sleep(1)
        print("starting")
        front_dis = 100
        left_dis =20
        right_dis =50
        front_dis = get_distance(TRIG_front,ECHO_front)
        left_dis = get_distance(TRIG_left,ECHO_left)
        right_dis = get_distance(TRIG_right,ECHO_right)
        print('Distance front: {} centimeters'.format(front_dis))
        print('Distance Left: {} centimeters'.format(left_dis))
```

```
        print('Distance right: {} centimeters'.format(right_dis))
        if obstacle_detected(front_dis):
            buzzer_beep()
            decide_direction(front_dis,left_dis,right_dis)
        if small_obstacle():
            buzzer_beep()
           play_sound("/home/pi/voices/small obstacles.wav")
        if moisture_check():
           buzzer_beep()
            play_sound("/home/pi/voices/water.wav")
        if check_darkness():
            LED_on()
        else:
            LED_off()

try:
    GPIO_setup()
    loop()
except KeyboardInterrupt:
    pass
finally:
    destroy()
```