



Waterford Institute *of* Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

# Embedded Systems Report - Lampbots

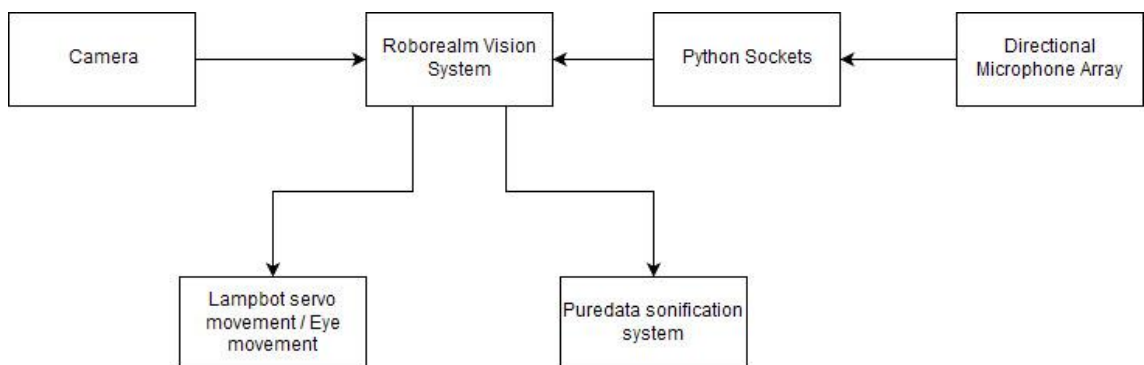
BACHELOR OF SCIENCE IN THE INTERNET OF THINGS  
YEAR 2 SEMESTER 4 - 2019  
ADAM COTTER - 20079973

## Table of contents:

1. Introduction – Page 1
2. Roborealm – Introduction and Tracking a Green Ball – Page 2
3. Roborealm – Face Tracking Version 1 – Page 5
4. Roborealm – Face Tracking Version 2 – Page 5
5. All Code – Page 10

## 1. Introduction

This report will cover my portion of the development of our class group project, the Lampbots. The lampbots were envisioned to be automated robots in the shape of desk lamps, equipped with moving joints, emotive eyes and vision and speech capacity using the microcontrollers we made in the previous semester. I was chosen to create a vision system for the robots.



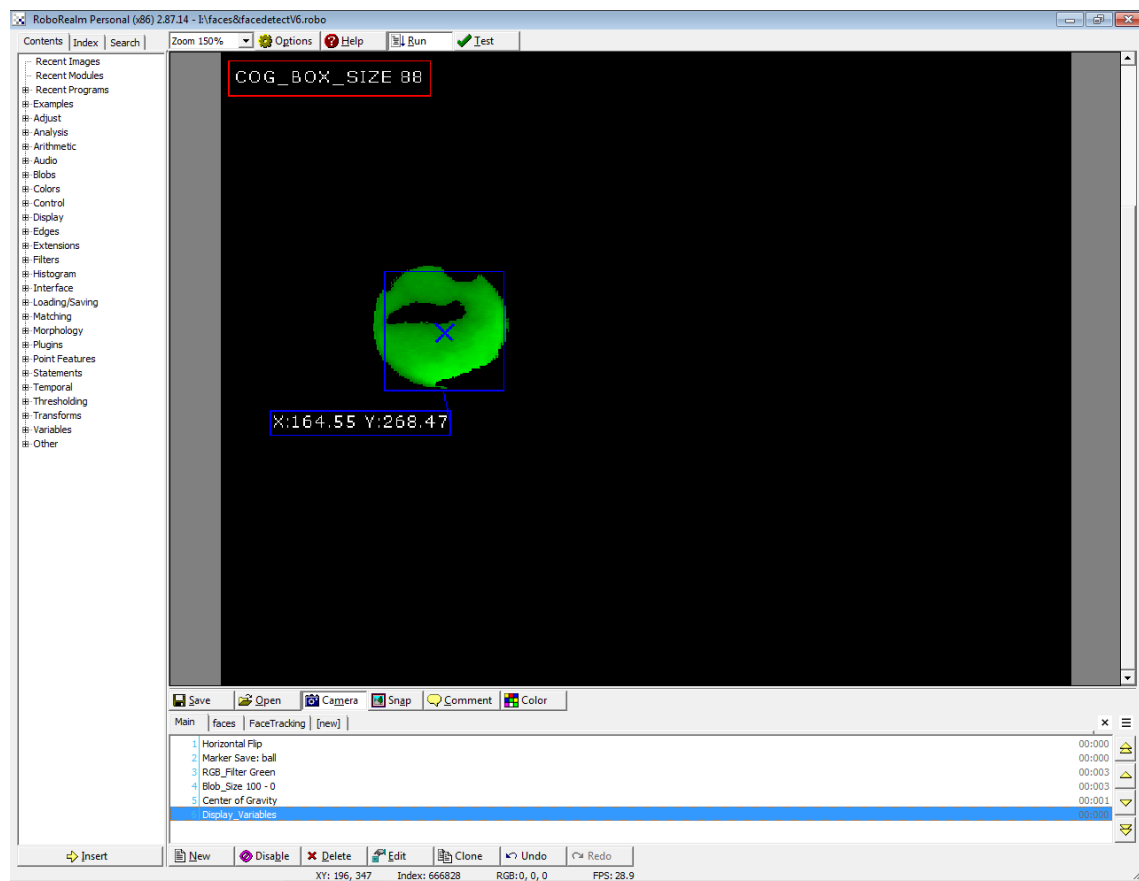
(Fig 1.1: Block diagram of full system)

Shown above is a diagram of all the inputs, outputs and systems in place to control the complete system.

## 2. Roborealm – Introduction and Tracking a Green Ball

The first step was to decide the software to be used for the vision system. Between Roborealm and OpenCV, Roborealm was chosen due to it being much more user friendly than OpenCV. Roborealm is a software system that allows users to create complex vision systems using a multitude of built-in modules and allows users to create custom modules using programming languages such as VBScript and C.

The first assignment was to create a program that tracked a green ball as a proof of concept for more advanced programs.



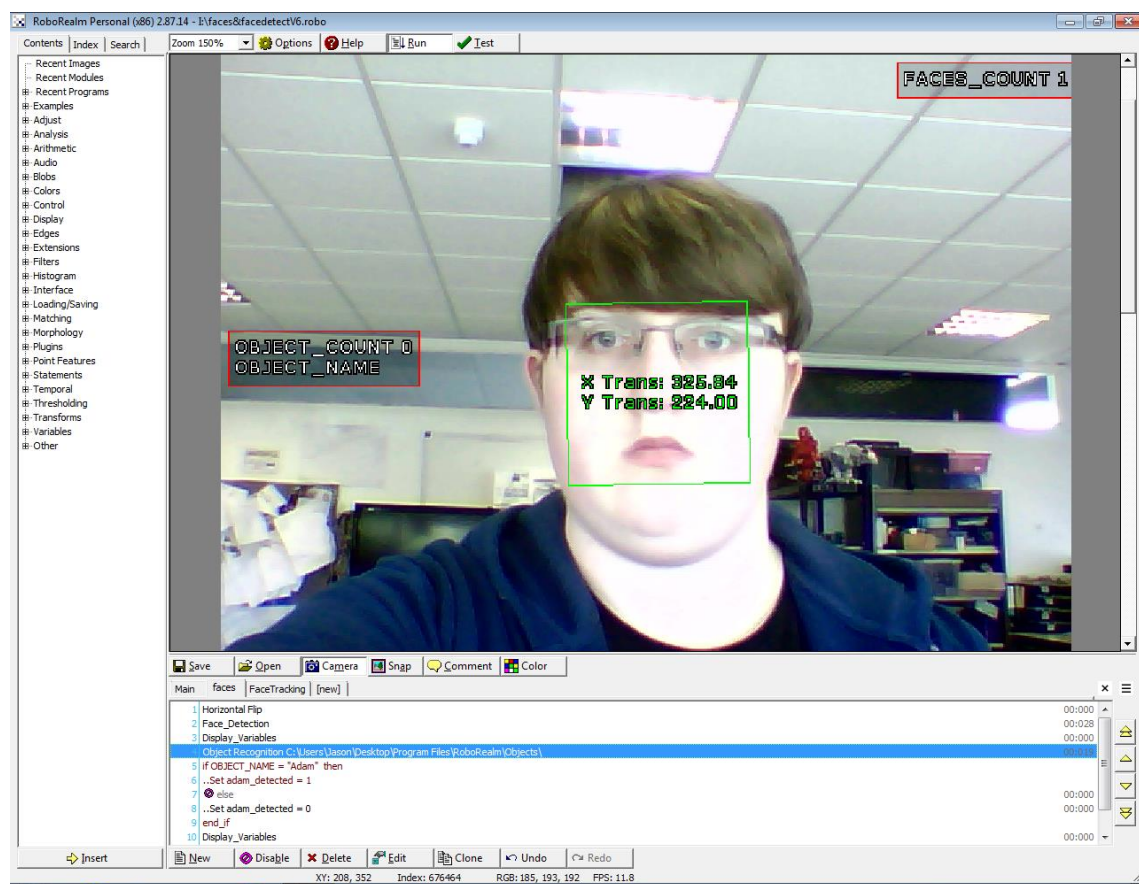
(Fig 2.1: Green ball tracking program)

The above program was created based on a program created by a previous student, Mark Dungan. The modules in the program do the following:

- **Horizontal Flip:** Flips the screen horizontally, resembling a mirror.
- **Marker Save ball:** Identifies a spot in the image.
- **RGB\_Filter\_Green:** Filters out all colours except green, allowing only the ball (or any other green objects) to be identified.
- **Blob\_Size 100-0:** This filters out all objects under a certain size, eliminating background noise.
- **Center of Gravity:** Gets the X and Y coordinate of the ball.
- **Display\_Variables:** Displays all the need variables on screen (X&Y coordinates).

### 3. Roborealm – Face Tracking Version 1

Next to do was to create a program capable of tracking faces.

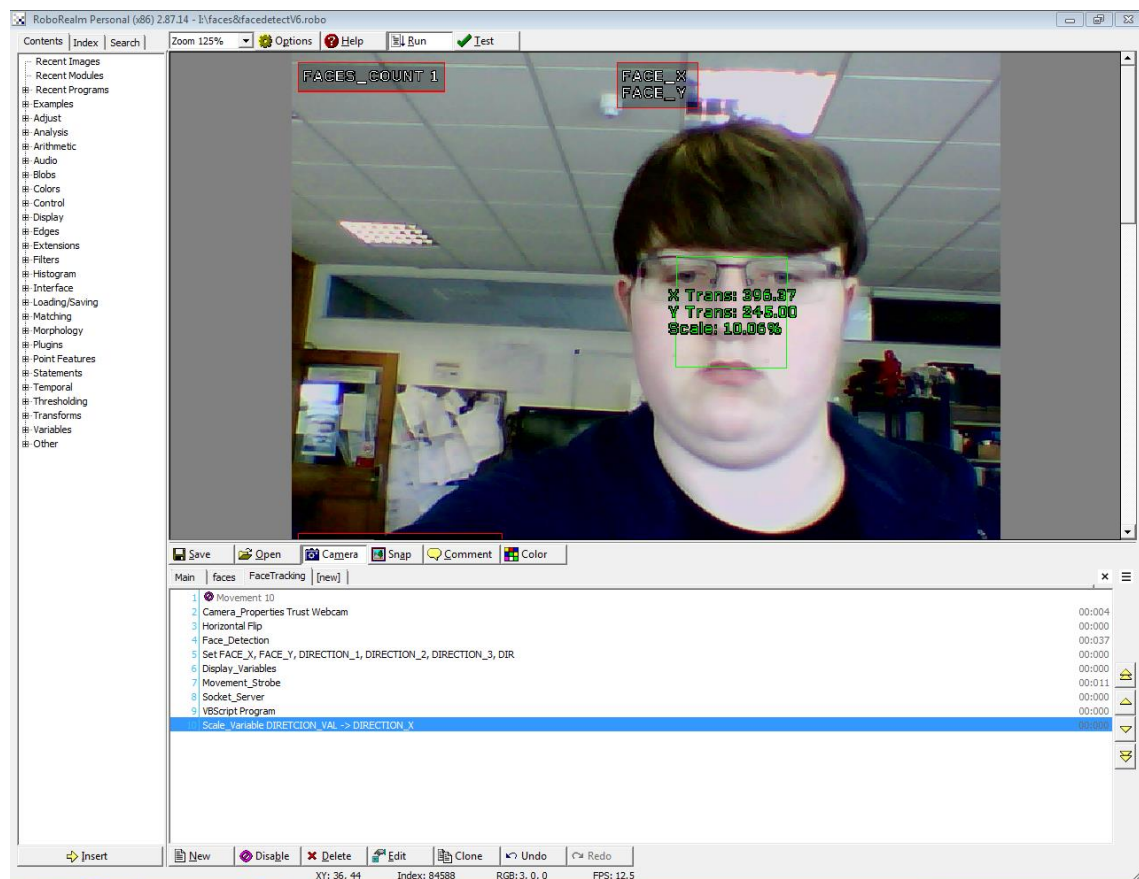


(Fig 3.1: First face tracking program)

The program shown above was the first version of a face tracking program created. It used the Roborealm “Face\_Detection” module to track any face that entered the view of the camera. In an attempt to get the program to recognise unique faces the “Object Recognition” module was used. Faces could be stored for the program as objects to be recognised. An if statement was set up to trigger when the program recognises the “Adam” face object (my face) and send information over OSC to a puredata program.

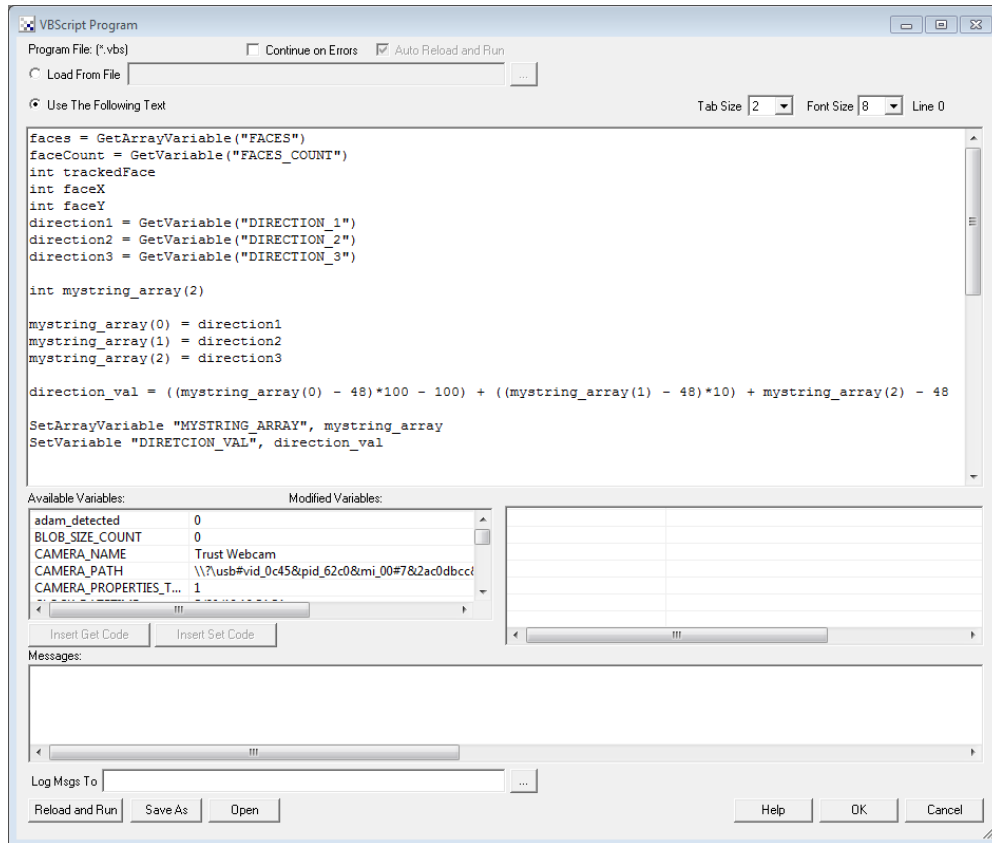
#### 4. Roborealm – Face Tracking Version 2

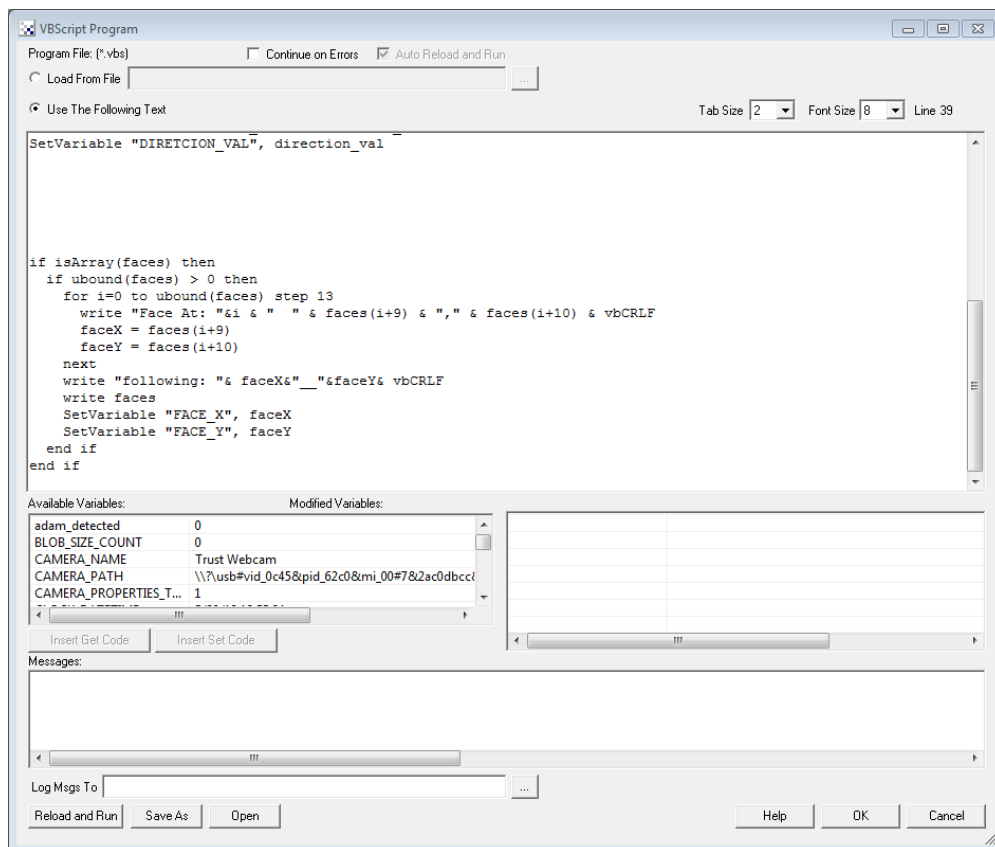
For the final version of the face tracking the unique face recognition was not used due to it being too complicated to implement with a large amount of faces. Instead the Face\_Detection module was only used.



(Fig 4.1: Final vision program)

This version of the program uses only the Face\_Detection module and uses a VBScript program to decide the most recent face to enter the camera's view.

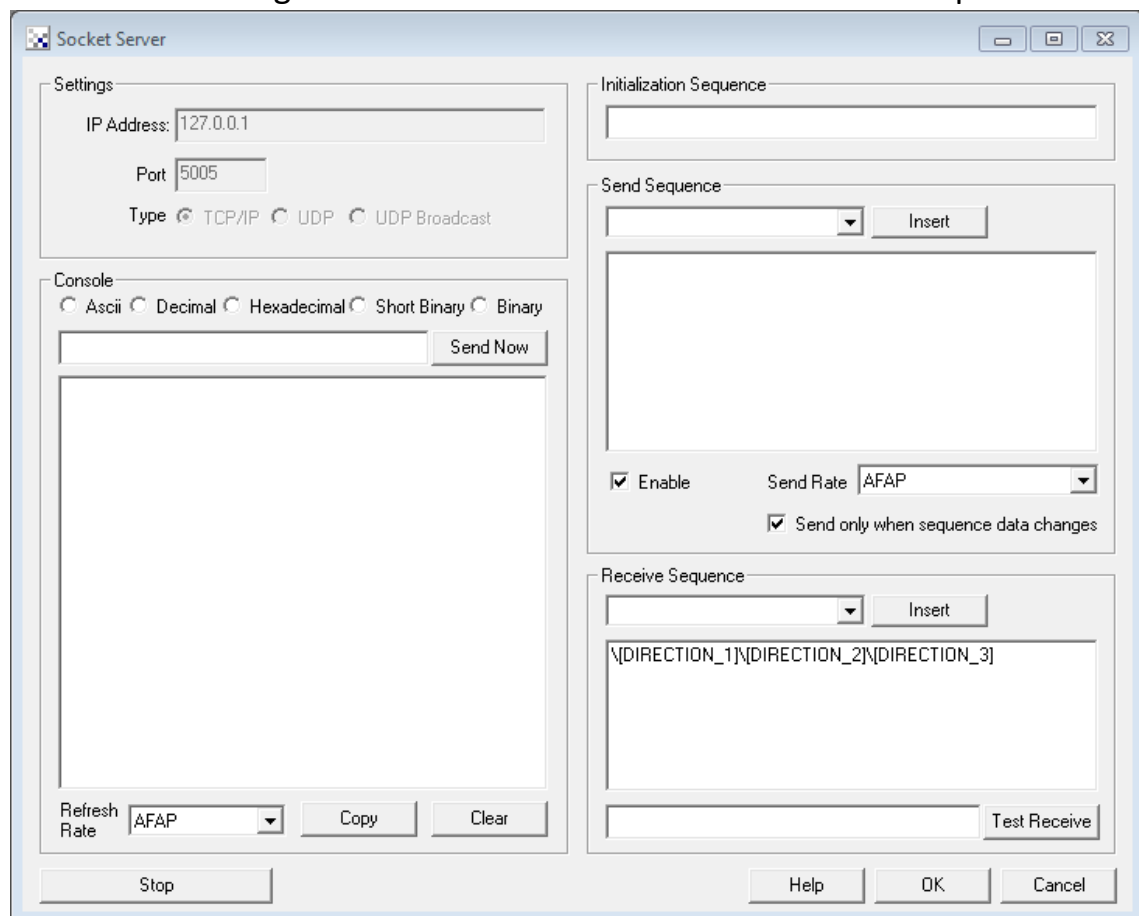




(Fig 4.2-4.3: The VBScript module)

Figure 4.3 shows the second part of the VBScript code, an if statement that sets the FACE\_X and FACE\_Y coordinate to the newest face added to the faces array.

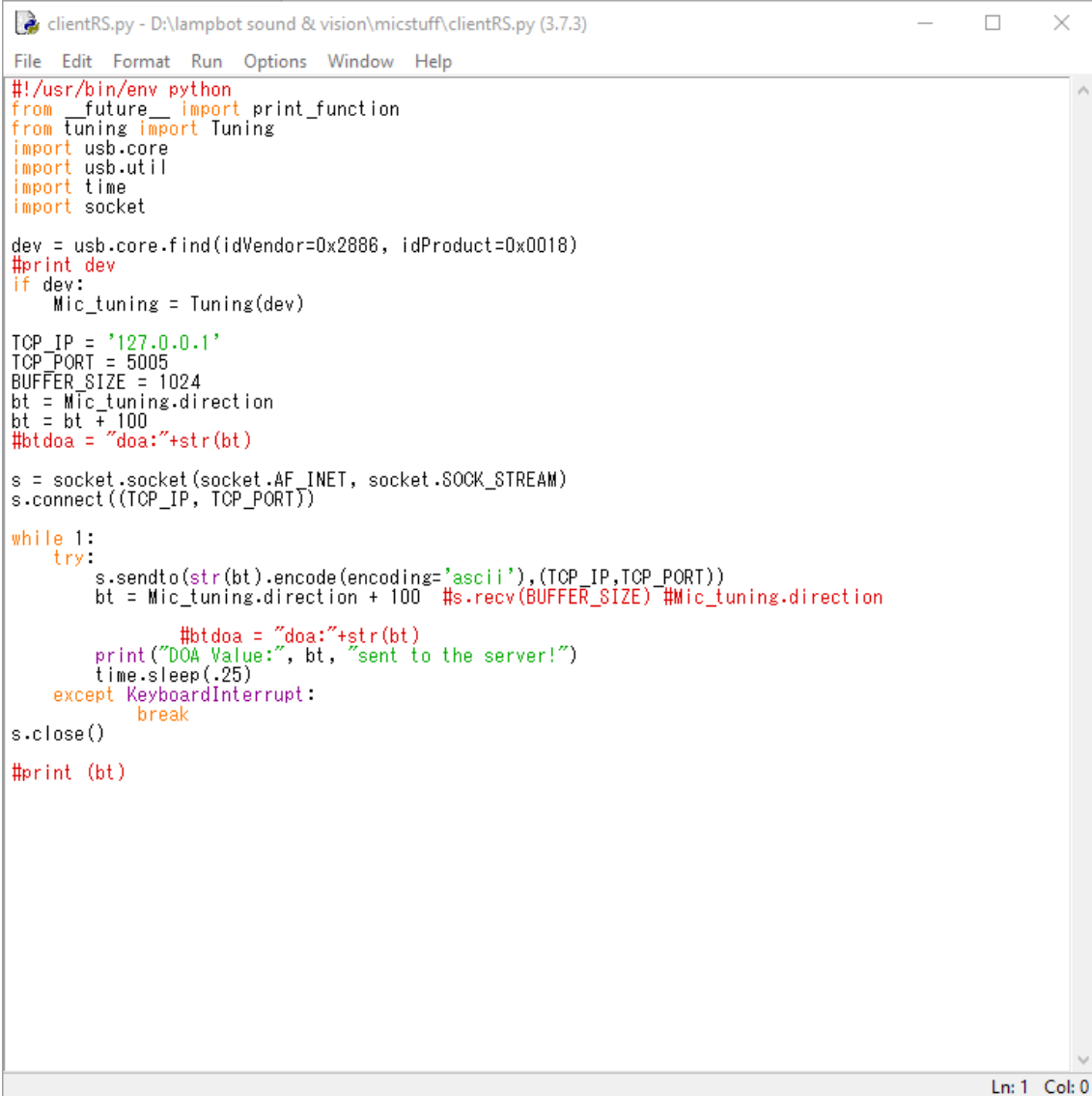
This program also has a socket server connected to a python script that sends the angle of a sound from a directional audio microphone.



(Fig 4.4: Socket server for listening the microphone script)

This socket server listens for data over the IP Address and port, parsing it in as 3 numbers, Direction\_1,2 and 3. The first part of the VBScript shown in figure 4.2 takes in these 3 direction values and carries out a calculation to scale the variable according to the camera's resolution. This was to have the Roboreal program take in the direction values from the microphone array, work out where on screen the sound originated, and have the robot look in the direction of the sound. Unfortunately this feature could not be implemented in time for project completion, but the values were successfully displayed on screen.





```
#!/usr/bin/env python
from __future__ import print_function
from tuning import Tuning
import usb.core
import usb.util
import time
import socket

dev = usb.core.find(idVendor=0x2886, idProduct=0x0018)
#print dev
if dev:
    Mic_tuning = Tuning(dev)

TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 1024
bt = Mic_tuning.direction
bt = bt + 100
#bt doa = "doa:"+str(bt)

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))

while 1:
    try:
        s.sendto(str(bt).encode(encoding='ascii'), (TCP_IP, TCP_PORT))
        bt = Mic_tuning.direction + 100 #s.recv(BUFFER_SIZE) #Mic_tuning.direction

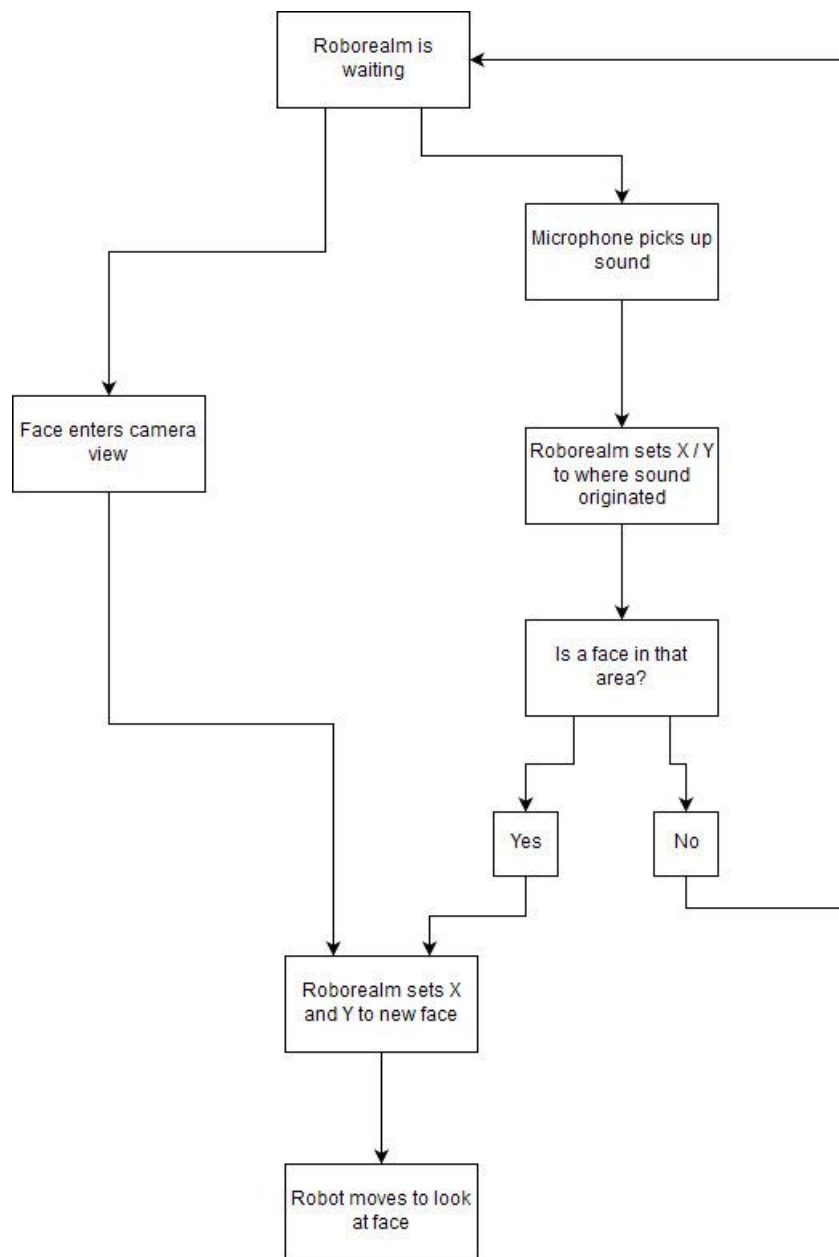
        #bt doa = "doa:"+str(bt)
        print("DOA Value:", bt, "sent to the server!")
        time.sleep(.25)
    except KeyboardInterrupt:
        break
s.close()

#print (bt)
```

Ln: 1 Col: 0

(Fig 4.5: Python client)

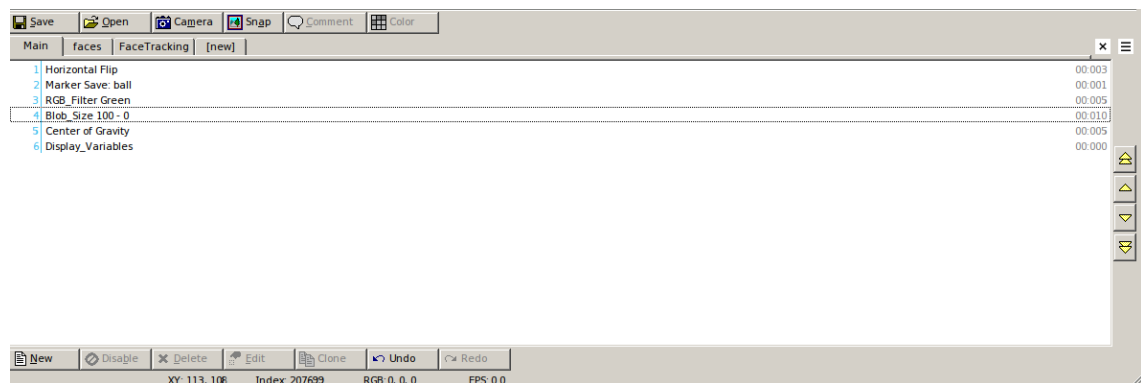
The script in figure 4.5 tunes the directional microphone and broadcasts it over the set IP address and port.



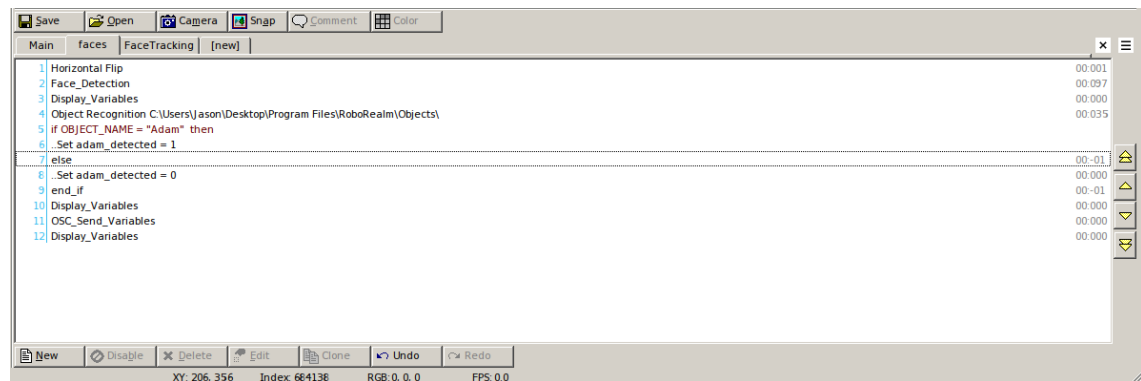
(Fig 4.6: Block diagram describing how the mic system works)

This version of the program was then given to the movement team in order for them to use the coordinates to move the body of the robot.

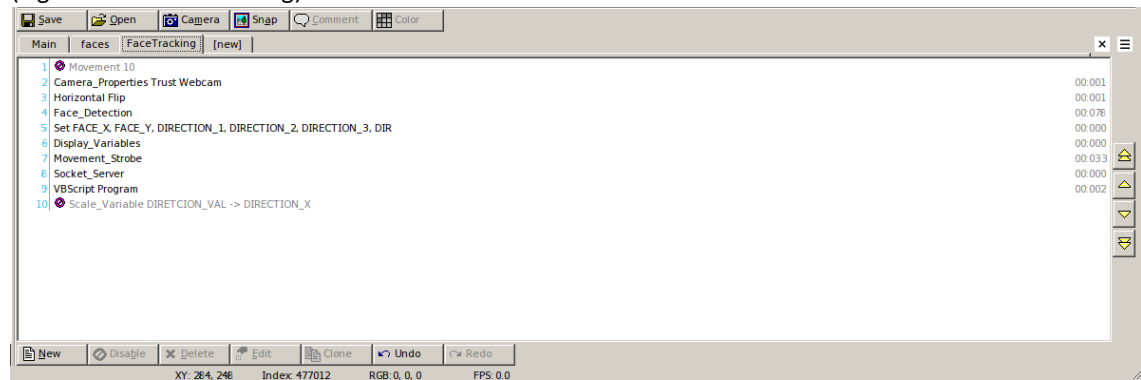
## 5. All Code



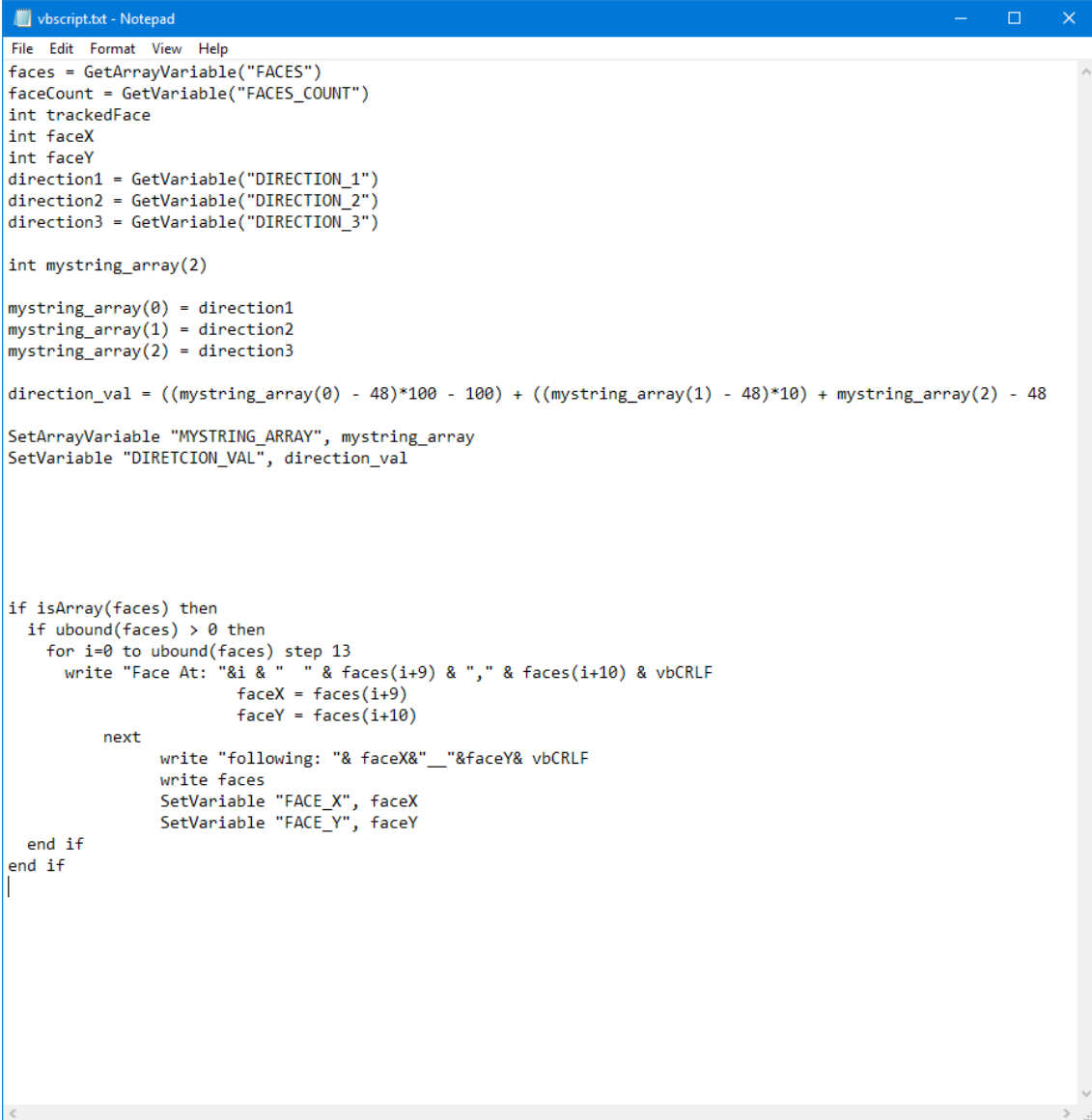
(Fig 5.1: Green ball tracking)



(Fig 5.2: First face tracking)



(Fig 5.3 Final face tracking)



```
File Edit Format View Help
faces = GetArrayVariable("FACES")
faceCount = GetVariable("FACES_COUNT")
int trackedFace
int faceX
int faceY
direction1 = GetVariable("DIRECTION_1")
direction2 = GetVariable("DIRECTION_2")
direction3 = GetVariable("DIRECTION_3")

int mystring_array(2)

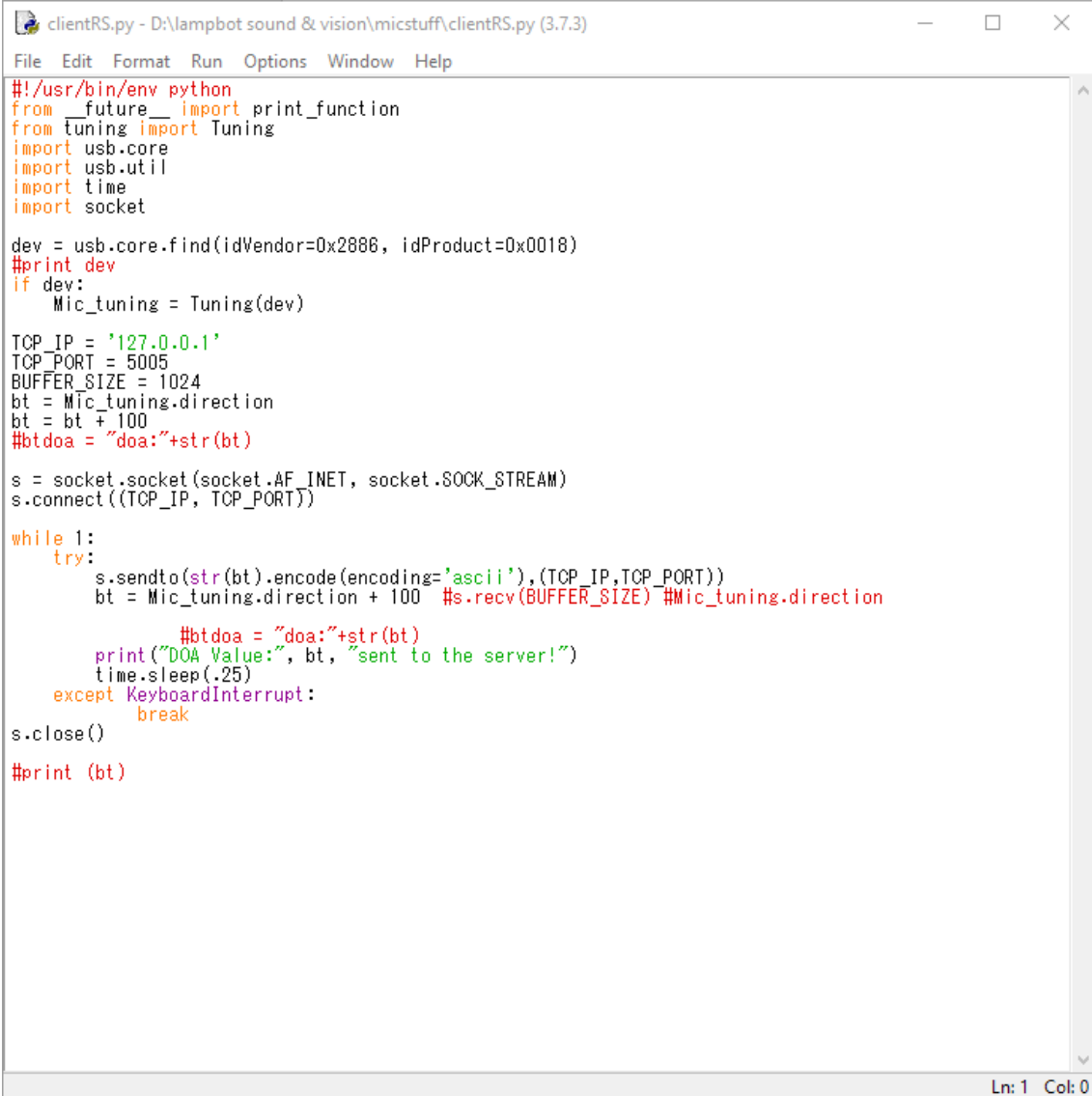
mystring_array(0) = direction1
mystring_array(1) = direction2
mystring_array(2) = direction3

direction_val = ((mystring_array(0) - 48)*100 - 100) + ((mystring_array(1) - 48)*10) + mystring_array(2) - 48

SetArrayVariable "MYSTRING_ARRAY", mystring_array
SetVariable "DIRETCION_VAL", direction_val

if isArray(faces) then
  if ubound(faces) > 0 then
    for i=0 to ubound(faces) step 13
      write "Face At: "&i & " " & faces(i+9) & "," & faces(i+10) & vbCRLF
      faceX = faces(i+9)
      faceY = faces(i+10)
    next
    write "following: "& faceX & "___"&faceY & vbCRLF
    write faces
    SetVariable "FACE_X", faceX
    SetVariable "FACE_Y", faceY
  end if
end if
|
```

(Fig 5.4: VBScript code)



```
clientRS.py - D:\lampbot sound & vision\micstuff\clientRS.py (3.7.3)
File Edit Format Run Options Window Help

#!/usr/bin/env python
from __future__ import print_function
from tuning import Tuning
import usb.core
import usb.util
import time
import socket

dev = usb.core.find(idVendor=0x2886, idProduct=0x0018)
#print dev
if dev:
    Mic_tuning = Tuning(dev)

TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 1024
bt = Mic_tuning.direction
bt = bt + 100
#bt doa = "doa:" + str(bt)

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))

while 1:
    try:
        s.sendto(str(bt).encode(encoding='ascii'), (TCP_IP, TCP_PORT))
        bt = Mic_tuning.direction + 100 #s.recv(BUFFER_SIZE) #Mic_tuning.direction

        #bt doa = "doa:" + str(bt)
        print("DOA Value:", bt, "sent to the server!")
        time.sleep(.25)
    except KeyboardInterrupt:
        break

s.close()

#print (bt)
```

Ln: 1 Col: 0

(Fig 5.5: Python client for microphone)