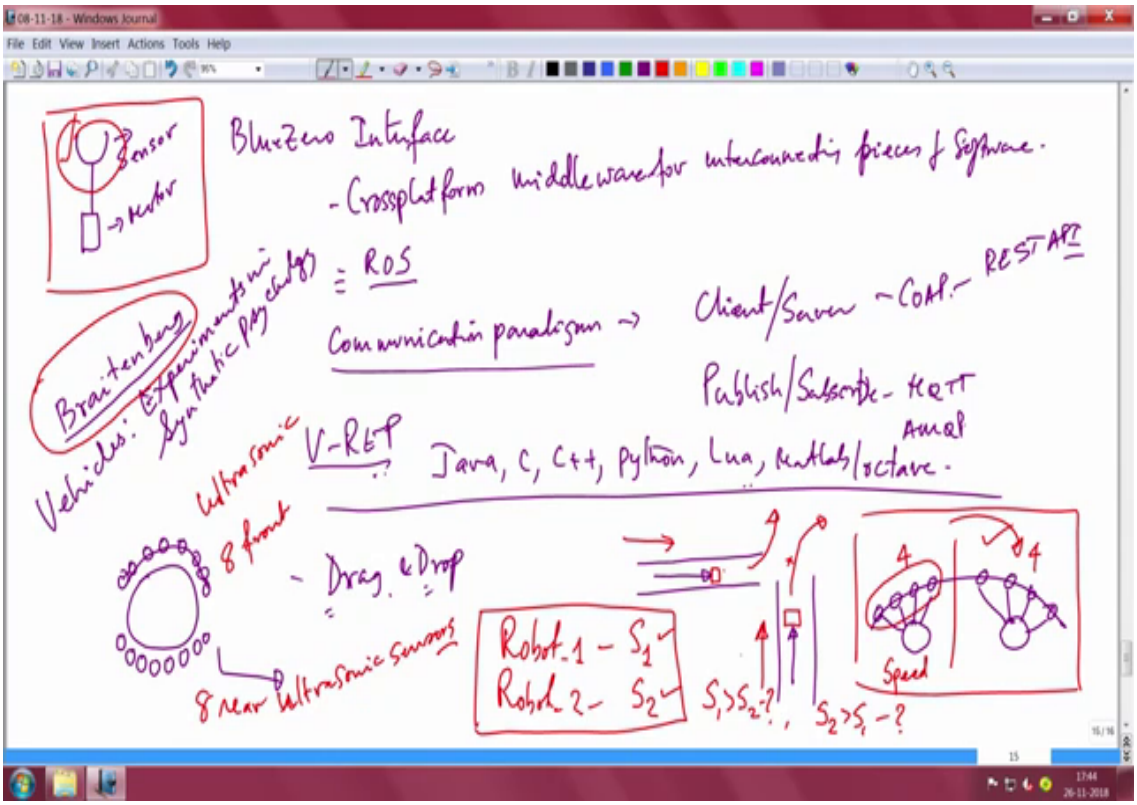
Advanced IOT Applications
Dr. T V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science, Bangalore

Lecture – 11
Building smart vehicle for collision avoidance

So, the example that you saw on V-REP and, the parallel we drew with the airplane example; means that there is an algorithm right that is what we said. Way back in the 1980s there was a book written by one person by name Braitenberg.

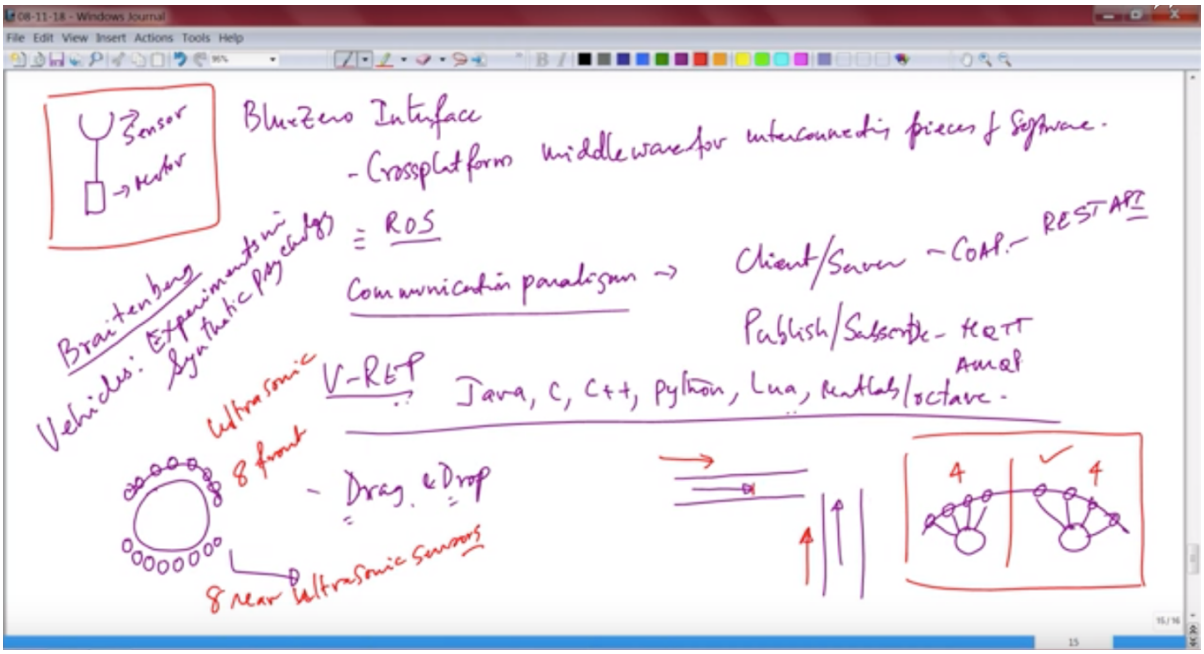
(Refer Slide Time: 00:52)



This researcher wrote a beautiful book and the book that he wrote is titled the vehicles experiments in synthetic psychology. He device many algorithms and many experiments and, one of the experiment is indeed the collision avoidance algorithm.

He denotes any sensor like a fork, you know the tuning fork right? So, his sensor is nothing, but a tuning fork. Any motor is essentially a like a rectangular box. In this particular example if you look carefully, the sensor is indeed the ultrasonic sensors and there are two motors. The sensor in the motor are directly coupled. This is a directly coupled system and

essentially we are trying to see how to use these sensors to control the motor? Sensor is giving some input and motor is doing something else how is this happening?



Now, the example that we took where the motor and there are a set of sensors as shown in the picture. So, let me take another colour to explain to you. How many sensors are there? There are 8 ultrasonic sensors in the front. And, then there are 8 rear ultrasonic sensors. So, what we have done is in this example, we have disabled all of them in the rear.

See the picture's left corner inside red box; There are 2 motors right our robotic platform in V-REP has 2 wheels each wheel is one motor. So, we associate the first 4 ultrasonic sensors to 1 wheel which is nothing, but one motor. And, the remaining 4 to the other motor or which is nothing, but the other wheel. So, you have all 4 together making it 8, this model you can imagine is very similar to what we have written here sensor and motor coupling.

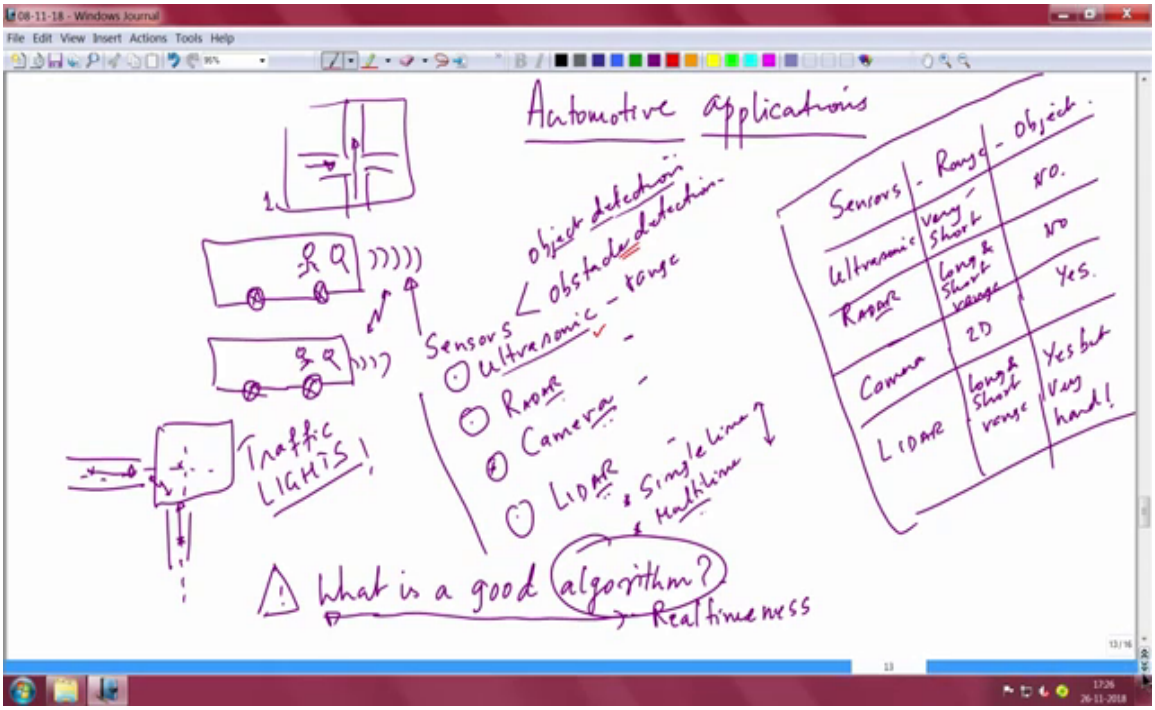
Now, concentrate on the example in the picture. What is the simplest algorithm that you can think of? You have two vehicle approaching each other from different direction towards an intersection to cross. Now, if communication is enabled between the 2 vehicles and sensing is enabled using ultrasonic sensors. I have a communication module which is part of the system that we have created, then in real time they know each one of these robots actually know their current location and at what speed they are approaching a particular intersection.

Now, once they realize that they are in very critical distance they can take decision about move, but how do they know they are in very critical distance? They know that because of this guy. The ultrasonic sensor tells them that they are extremely close, there is a threshold a simple threshold based system is applied. If they are within critical threshold distance the 2 robots decide to go in 2 different directions and that is easy to envisage like this; if there is an object approaching from left, the left sensors will increase the speed of left motor and if the object is approaching from right, the right sensor connected to right motor will increase the speed, which makes the robot move away from the object approaching it.

The same thing will happen to the other robo as well, the moment it senses the object it will move away in the other direction. Who proposed this direct coupling between sensor, motor, and all of that it is all done by the very simple example and this is nothing, but the simplest braitenberg algorithm, that has been coded within the V-REP to show you this demonstration.

Now, it is so, simple, but very intuitive; you can use this as a starting point to build very very complex algorithms to do a collision avoidance. Now, different type of matrix can be adopted for distance calculation. One is you can do simple square root of x square plus y square plus z square, if you know the positions you know the distance estimation of the 2 robot systems or you can use other type of distance estimations. So, in the sense Braitenberg algorithm is essentially trying to has been coded in this and it gives you a intuitive example of how you can do collision avoidance.

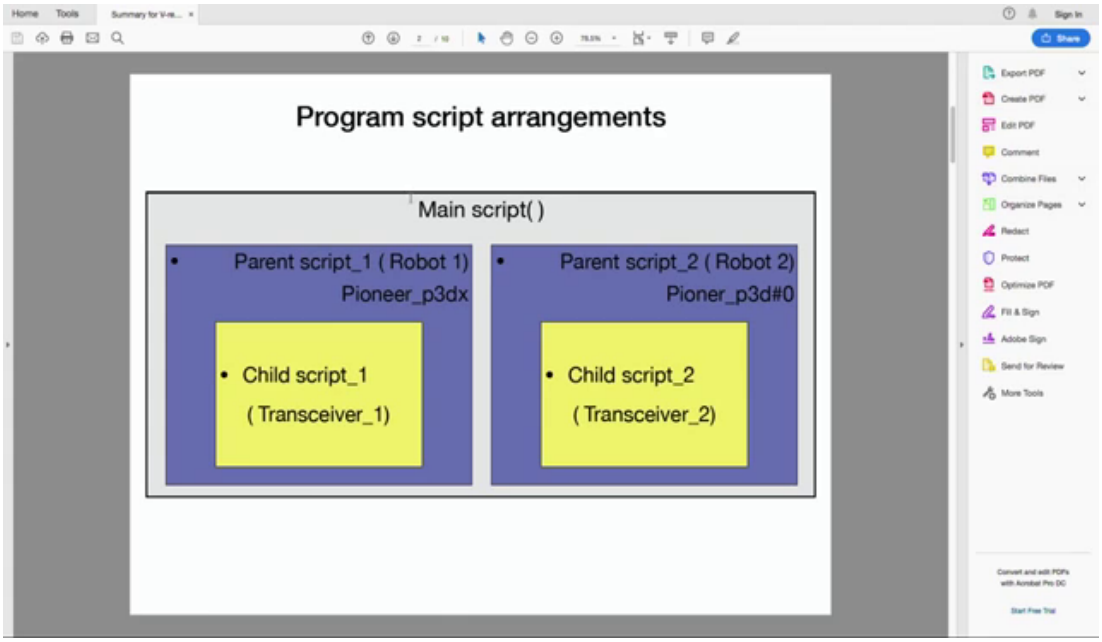
(Refer Slide Time: 09:06)



Remember what we said very critically about obstacle avoidance ? that you have to do object detection and then you have to do obstacle detection. So, unless you detect an object you cant do an obstacle detection. All of this means you need sensors and here are in this case we are using ultrasonic sensors to essentially detect that there is an obstacle in close proximity, and then we are moving away from each other using a very simple braitenberg algorithm right?

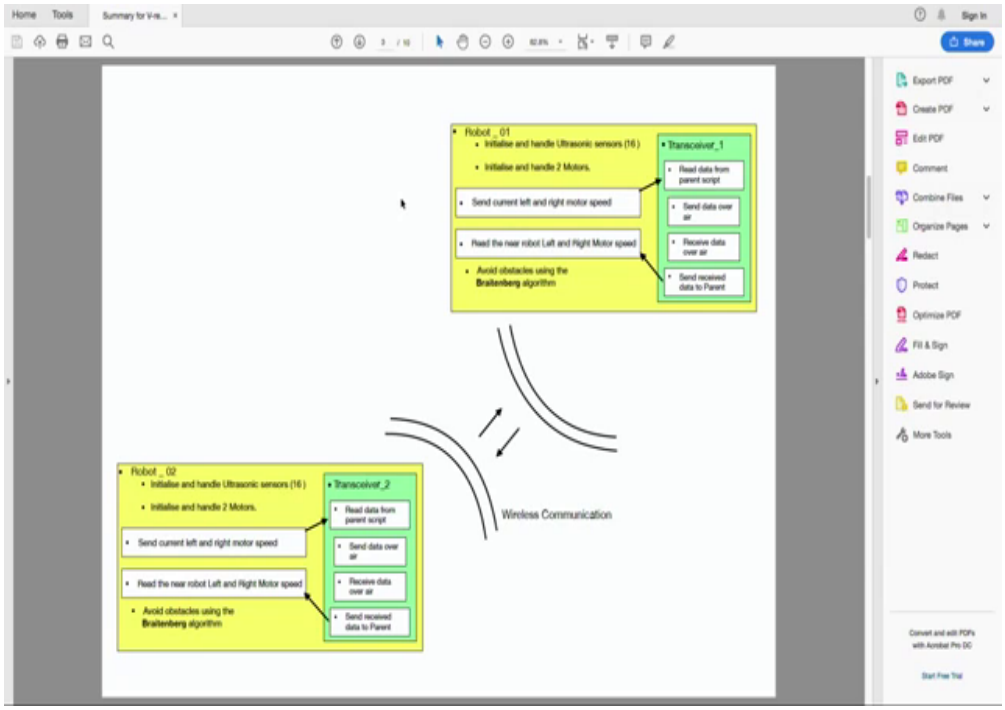
So, let us capture all of this in terms of some nice high level view, which will help you to connect even better on this example.

(Refer Slide Time: 10:02)



Let us now turn our attention to the program scripts here. This is essentially what is being coded. There is a main script, you know that there are 2 robots; robot one and robot 2. We have taken a model called pioneer underscore p 3 dx to which is associated a transceiver module placed on top of it which becomes a child. So, the parent will have a script and the child will have another script.

(Refer Slide Time: 10:38)



So, essentially there are 2 pieces of code at which are working together. The Braitenberg algorithm that you see is quite straight forward I hope you can read the slides quite well.

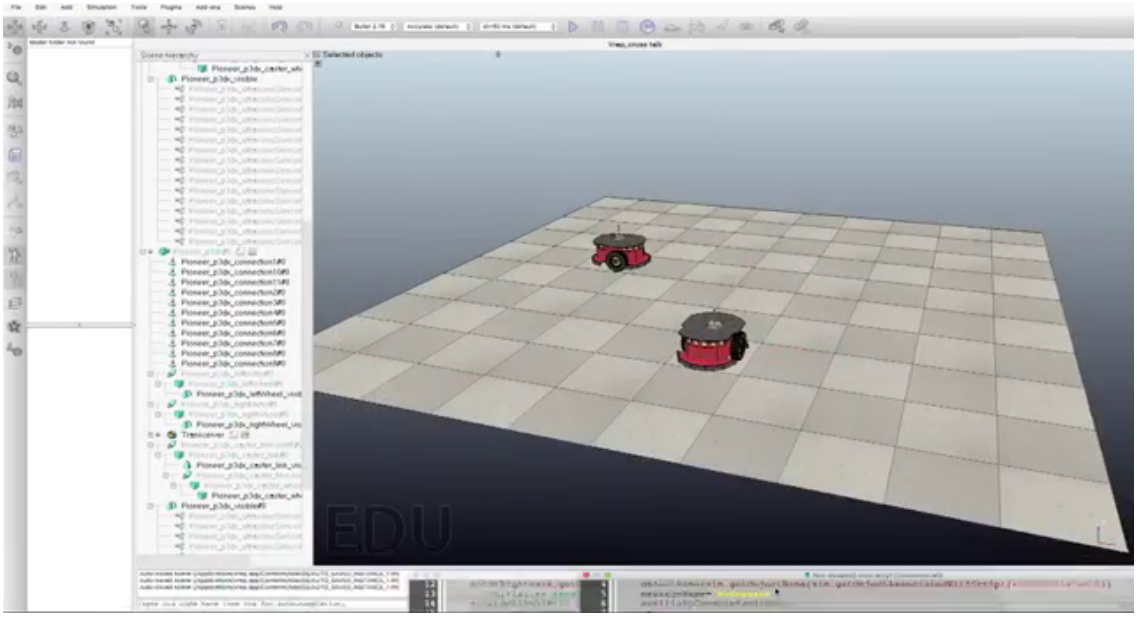
Take robot 2 let us start with robot 2, initialise and handle ultrasonic sensors 16 disable 8 of them not shown here, but you have to initialise all of them because you have taken that model. Initialise and handle 2 motors our robo has only 2 model 2 motors, send current left and right motor speed to whom to it is child. So, that it can convey that information to the other robo. So, read data from parent script; that means, it is extracting this information from the transceiver module. And, it is constantly sending the data over the air, whether there is another robot or not is not of importance continuously keeps transmitting as a beacon goes on sending out the current left and right motor speed being conveyed out on the air.

Now, if there is indeed an other robo in it is communication range, it receives the data over the air and then it does some processing and sends it to the back to the parent. And, the parent now says I will now read the near robot left and right motor speed, because it is coming from the other remote side. And, then avoid obstacles using the braitenberg algorithm, which is essentially using the ultrasonic sensor and a certain threshold applied to the range.

And, all of this is going through a communication system as you can see it is the communication medium is wireless, and the same replication actually happens on the other robo as well initialise, and then handle the 2 motors send the current location current to the left and right motor speed, and then read the near robot left and right motor speed from the remote side other robots, the left and right motor speed.

Then, also run the same algorithm like 2 airplanes at cruising speed right they are just avoid they have a collision avoidance algorithm. So, quite like that all right.

(Refer Slide Time: 13:19)



So, that is interesting part and now it would be useful to just run it once to connect everything together, there you are they are coming closer and you have seen a completely different demonstration now right. This is your lab exercise how did this happen? In this experiment see an interesting thing happens can we run it again. So, there you are the other robo actually has stopped and then allowed the other robot to go away, which is very very interesting. Why do not you try this? This you should try as a lab exercise for yourself to learn a little more.

This is another type of obstacle avoidance right, it is also trying to do obstacle avoidance, but of a different type. It is not parting ways or it is not moving away in 2 different directions, but instead it is stopping and allowing the other robot to pass after which it decides to pass. The left side robo as you can see waits for the right side robo to pass, let

us try that algorithm again look carefully. So, it now quite simple for you to make modifications to this code, what is actually happening?

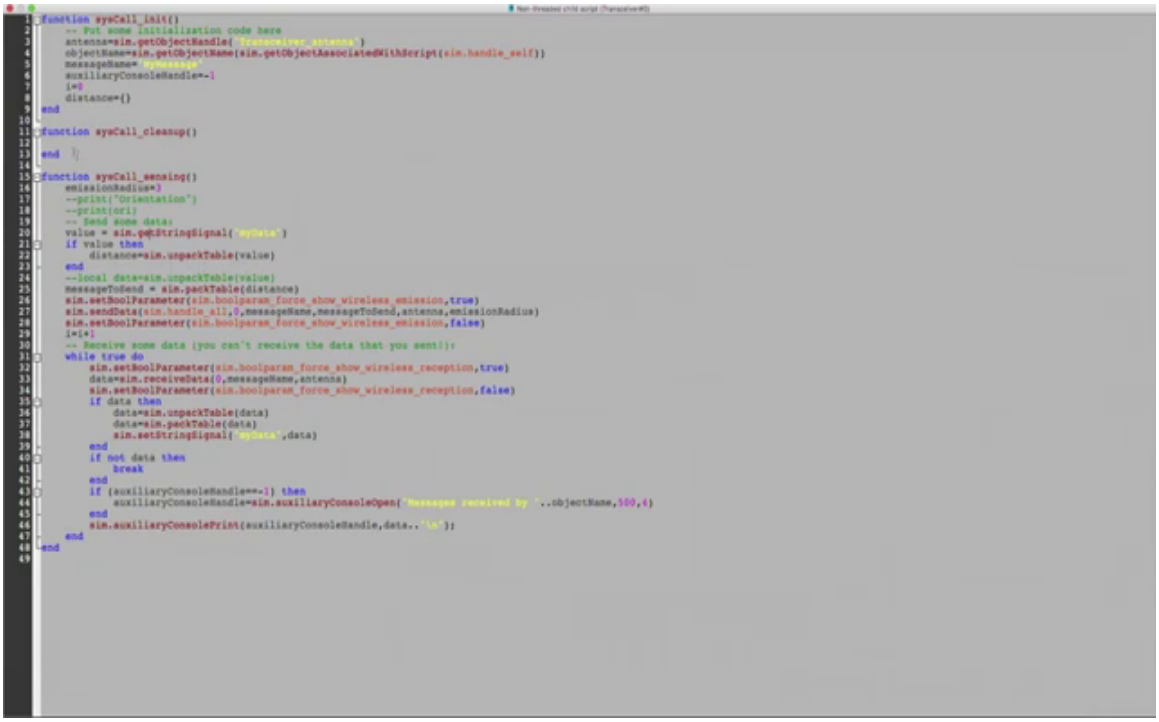
Now, you can easily modify the following. So, what you should do is, it is quite simple quite intuitive the robo here has an ultrasonic sensor; the robo here has a set of sensors. In the first case what we did was, we ran the braitenberg algorithm to do obstacle avoidance. In this case if they are in critical distance, you just stop do not run the braitenberg algorithm that's all. You disable the braitenberg algorithm and, but if you disable the braitenberg algorithm there is a likelihood that they will collide.

So, how you avoid the collision? You have to do some more glue logic you have do something interesting, you have to by and large use the ultrasonic sensor to say that, if I move a little more forward than what the ultrasonic sensor says I am going to collide and therefore, I decide to halt.

Now, both may decide to halt right. In this case you can see in this demo, we have coded in a manner that only one halts. Now, you have to take a call, whether it should be the left side one that should halt or the right one that should halt. That is a matter of how you code everything? In this case I can give you a hint that whoever is slow should halt, that is all you need to put the glue logic. It appears if you run the demo again you might not see it directly, but you will see that indeed the left side robo was moving a little slow compare to the one that is moving away, moving straight away, which gave it the a priority to do an intersection crossing because it was moving at a higher speed.

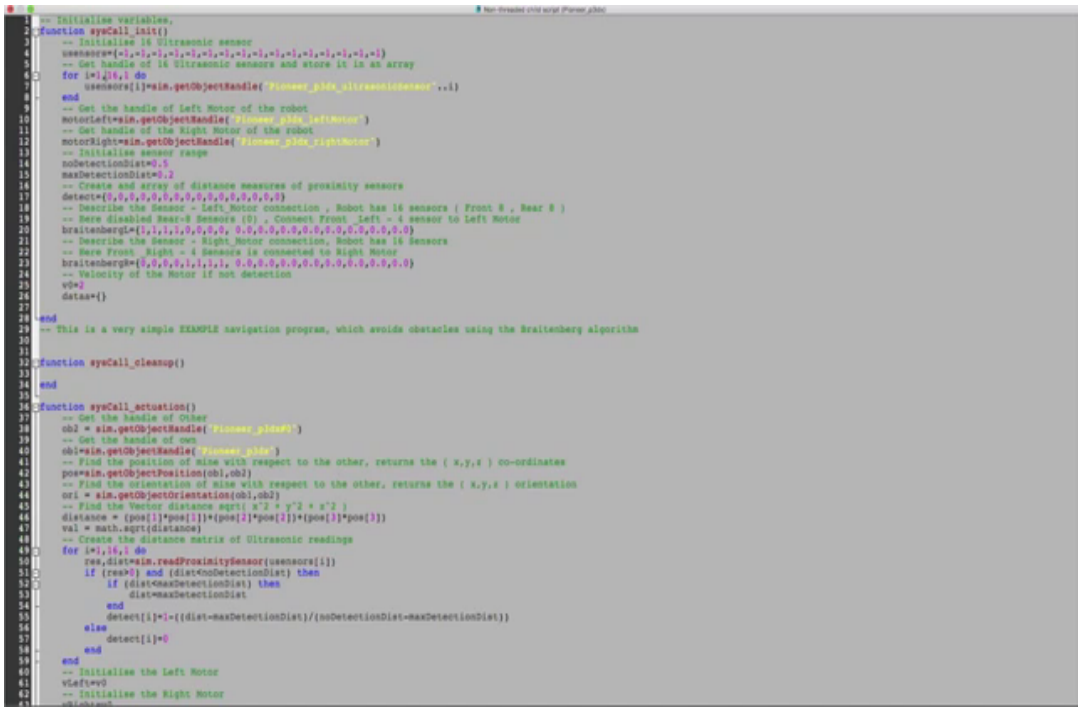
So, what you do now you take this algorithm - basic code which are going to be available to you apply this glue logic, which essentially says I will use ultrasonic sensors to do thresholding I will put a threshold on it, and moment that threshold happens, and I know that I am slow I have to halt so, that I can let the other robo pass. This is all that would be required for you to try and experiment it. Of course, I am not going to leave you high and high let us try and understand this code, we will run through this code for you.

(Refer Slide Time: 17:12)



But, what will be available on the internet or from the, for you to try would be the basic code. All though we will show you here it is important that you try it yourself in a effective manner.

(Refer Slide Time: 17:23)

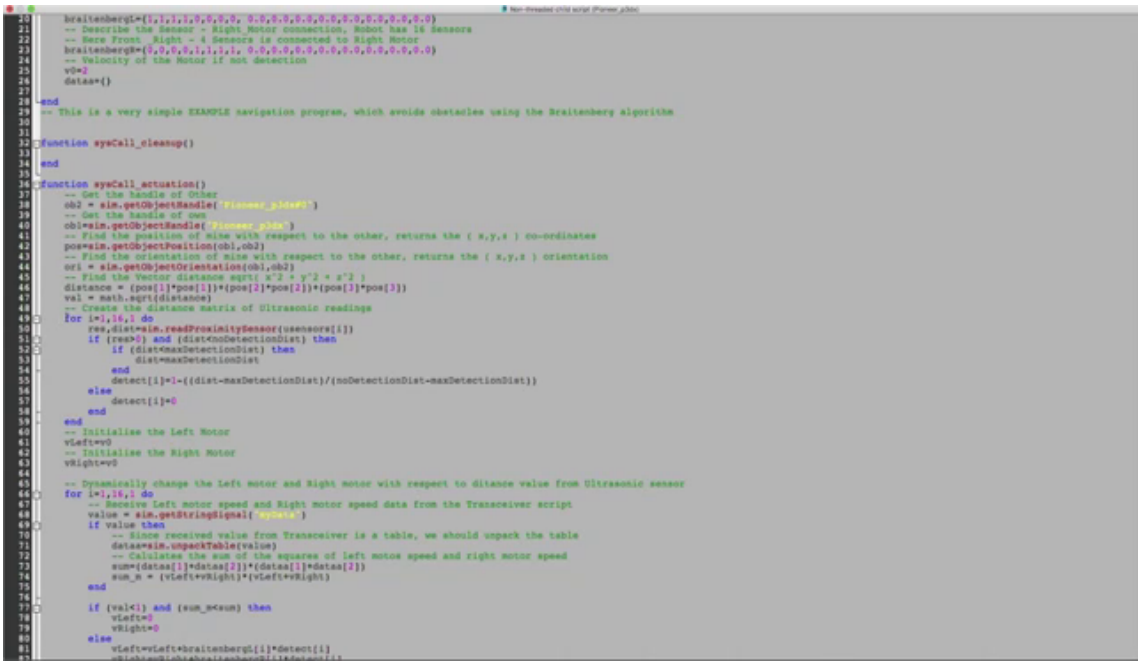


So, first step would be initialising the variables. So, as you can see that is initialise the 16 ultrasonic sensors, there are 8 in the front and there are 8 in the back and making 16 and all that. You are using only 8 that we mentioned this already, but you have still initialise all of them, get the handle for the left motor and the right motor that is also shown there and then you initialise the sensor range. So, this is an important step right. So, you have to initialise the sensor range.

Then you create and an array of distance measures of proximity sensors, you can see that detection is all 0; that means, you are initialising it all of them to 0. And, what is important is to ensure that, you are using only the 4 for the left. you can see they are all 1 1 1 1 these are binary right; 1 means it is initialised, 0 means they are not used. So, it is just a binary decision. The left 1 uses 4 and the right one Braitenberg_r uses the other 4, the rest of it are all 0s so, which is the clear indication, that they are not being used all right.

Then this is so, that is about what we are doing?

(Refer Slide Time: 18:59)



Then let us go to the next step is to get the handle of the other of course, there are 2 robots. So, both the robots have to essentially work in tandem. So, now, let us further look at the code. Here you can see that the way the V-REP codes it is you have to understand get use to the way it codes. Essentially what is sent over the air is the motor speed right that is all

you are sending. And, from the motor speed you are now computing the position x y z of this robo, or my position relative to the other relative to the other robo.

So, this is a continuous process each time you are computing based on the motor speed, what is my position, current position with respect to or with relation to the other robots position.

So, this both robots are doing this computation and obviously, this distance is coming down as they are coming closer and closer. And, then they also note their speeds, because motor speed is directly related to the speed of the vehicle, motor speeds are known those speed of the 2 robots are known to each other. Now, let us say one robot speed is S 1 and for clarity let me put it here right. So, you can see here this is robot ones speed is S_1 and robot 2s speed is S_2.

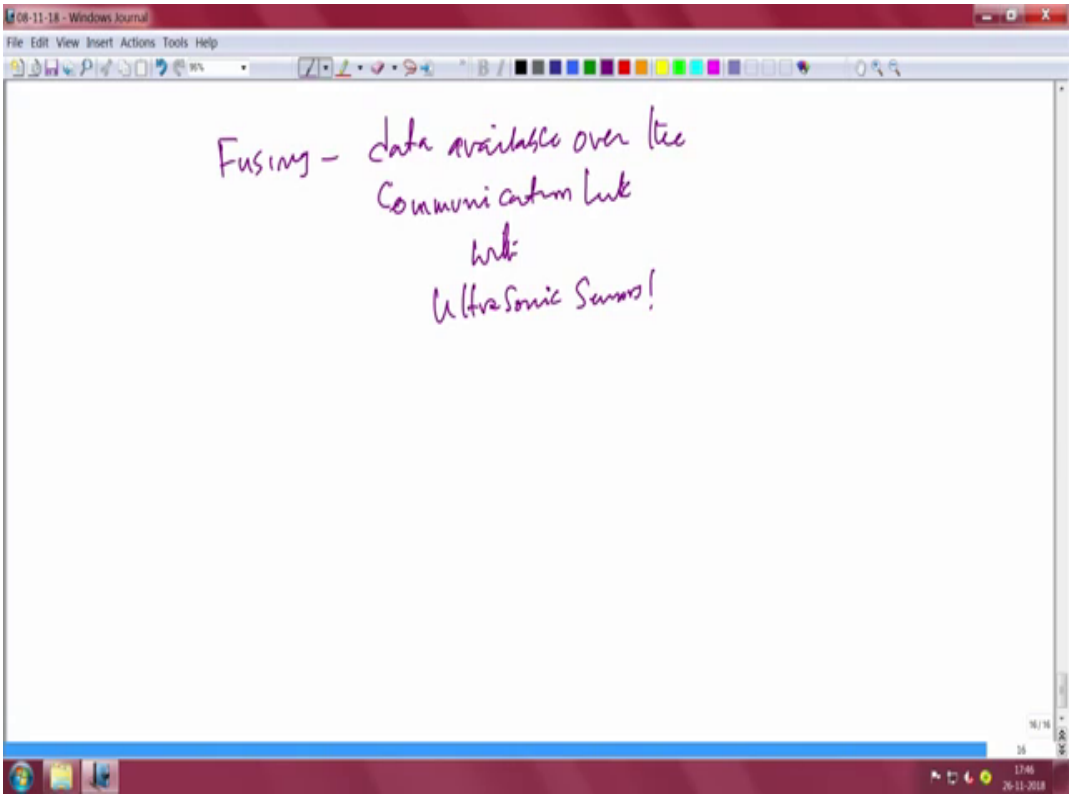
Now, you can apply a simple logic is S1 greater than S2 what should you do? If S2 is greater than S1, what you should do? Both are possible. Now, that you are computing. So, this is very critical. So, let us look back at this code. And, this code is saying I am continuously evaluating my xyz position relative to the other. And, I am also monitoring the ultrasonic sensors are we coming closer to each other that part is shown exactly here, you can see that the ultrasonic part is out there click on that distance create the distance matrix of ultrasonic readings. So, that is also there.

(Refer Slide Time: 21:49)

```
function sysCall_cleanup()

end

function sysCall_actuation()
    -- Get the handle of Other
    ob2 = sim.getObjectHandle('Pioneer_p3dx#0')
    -- Get the handle of own
    ob1=sim.getObjectHandle('Pioneer_p3dx')
    -- Find the position of mine with respect to the other, returns the ( x,y,z ) co-ordinates
    pos=sim.getObjectPosition(ob1,ob2)
    -- Find the orientation of mine with respect to the other, returns the ( x,y,z ) orientation
    ori = sim.getObjectOrientation(ob1,ob2)
    -- Find the Vector distance sqrt( x^2 + y^2 + z^2 )
    distance = (pos[1]*pos[1])+(pos[2]*pos[2])+(pos[3]*pos[3])
    val = math.sqrt(distance)
    -- Create the distance matrix of Ultrasonic readings
    for i=1,16,1 do
        res,dist=sim.readProximitySensor(usensors[i])
        if (res>0) and (dist<noDetectionDist) then
            if (dist<maxDetectionDist) then
                dist=maxDetectionDist
            end
            detect[i]=1-((dist-maxDetectionDist)/(noDetectionDist-maxDetectionDist))
        else
            detect[i]=0
        end
    end
    -- Initialise the Left Motor
    vLeft=v0
    -- Initialise the Right Motor
    vRight=v0

    -- Dynamically change the Left motor and Right motor with respect to ditance value from Ultrasonic sensor
    for i=1,16,1 do
        -- Receive Left motor speed and Right motor speed data from the Transceiver script
        value = sim.getStringSignal('mydata')
        if value then
            -- Since received value from Transceiver is a table, we should unpack the table
            datas=sim.unpackTable(value)
            -- Calculate the sum of the squares of left motor speed and right motor speed
            sum=(datas[1]*datas[2])*(datas[1]*datas[2])
            sum_m = (vLeft*vRight)*(vLeft*vRight)
        end

        if (val<1) and (sum_m<sum) then
            vLeft=0
            vRight=0
        else
            vLeft=vLeft+braitenbergL[i]*detect[i]
            vRight=vRight+braitenbergR[i]*detect[i]
        end

    end
    local data=sim.packTable({vLeft,vRight})
    sim.setStringSignal('mydata',data)
    sim.setJointTargetVelocity(motorLeft,vLeft)
    sim.setJointTargetVelocity(motorRight,vRight)
end
```

And, then once they realize that the let us say robot 1 is slower compare to robot 2 speed wise. So, in other words let us say S1 this condition, S2 is greater than S 1 that is the speed of robot 2 is higher compare to robot 1. Then, what it does is robot 1 decides to cut speed and actually come to a dead halt soon as the ultrasonic sensor says that you are in critical distance. So, the way to fuse so, it is actually a fusing method.

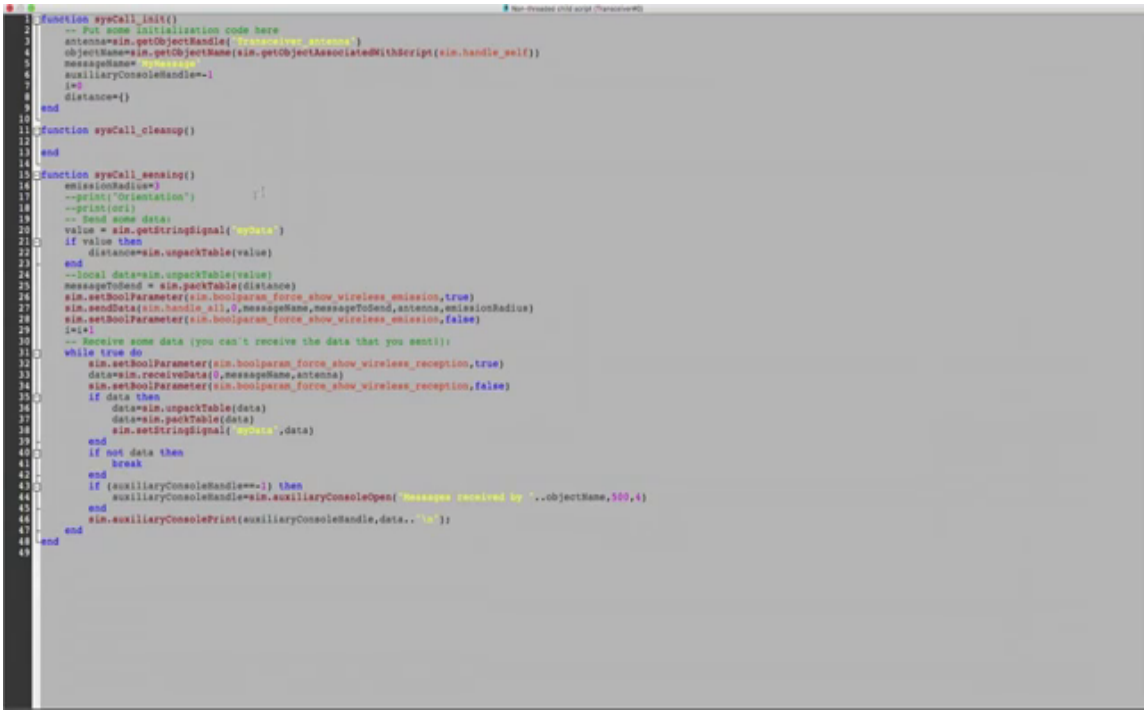
(Refer Slide Time: 22:32)



So, what are you doing? You are fusing the data available over the communication link with the ultrasonic sensors. And, then taking an intelligent decision to halt robot 1 and let robot 2 pass and that is exactly the demo that we had seen.

So, this is in brief what you should be doing you should be coding and all of that you can see is that logic essentially is shown here. If, there is an if else condition which is essentially accomplishing that step right. So, let us look at the transceiver script which is also pretty straight forward.

(Refer Slide Time: 23:45)



This is what the transceiver does, what should it do? It should take data from the parent pass it out on the air interface it should; obviously, data is available in a particular format. So, it should pack and unpack and again pack and unpack. So, essentially marshalling and un marshalling the data has to happen. So, it is doing exactly this on the one side it takes it from the parent and puts it out on the air interface. After suitable packing and once it receives data from the air interface, it does all the unpacking part. And, then puts it gives it back to the parent and then let us parent let the parent take a decision right, that is all this code is actually doing.

So, this piece of code the child script and the parent script together essentially will allow you to put this demonstration in place. You can start with the basic code available on the V-REP which is nothing, but the Braitenberg algorithm for which you do not have to code anything it will work directly. And, if you have to modify it you have to do this little logic of checking the speeds and then taking a decision on which one should halt and let the other robot move forward.

Thank you.