Prototyping and Systems documentation (GROUP B6)

Kuye doluwamu Taiwo
Electronic engineering
University of applied sciences hamm lippsdadt
Lippstadt,Germany
doluwamu-taiwo.kuye@stud.hshl.de

Ashrafuzzaman siddiqi mohammad
Electronic engineering
University of applied sciences hamm lippsdadt
Lippstadt,Germany
mohammad-ashrafuzzaman.siddiqi@stud.hshl.de

Yashodhan vishvesh Deshpande
Electronic engineering
University of applied sciences hamm lippsdadt
Lippstadt,Germany
yahsodhan-vishvesh.deshpande@stud.hshl.de

***Abstract*— This document explains our methodology and work during the prototyping and systems engineering courses. It includes all of the methods and systems we used to accomplish the smart farming vehicle's construction. (*Abstract*)**

Keywords—models,activity,design,Arduino,code

I. Introduction

At the start of this prototyping and systems engineering course we were tasked with designing and developing a vehicle for smart farming, in the first few weeks we attended lectures on mostly systems engineering, where we learnt about control engineering software engineering and the relation between them, we also learnt about requirement engineering and its importance we were also taught various UML and SYSML modelling languages. Our project consists of 3 major parts the system engineering part the Design part and the Programming part.

II. Systems engineering

We learned about requirement engineering and models in the system engineering section of this course, as well as how to utilize these tools and apply them as engineers, in the first iteration of our project, we had to model the planned vehicle and system and our system requirements.

A. First Iteration of project

The first iteration of the project was conducted by Professor Dr Henkler we modelled diagrams for a smart farming vehicle use case and conducted a practical with an already made vehicle.

1) Requirement diagram

a) 1st iteration of requirement diagram

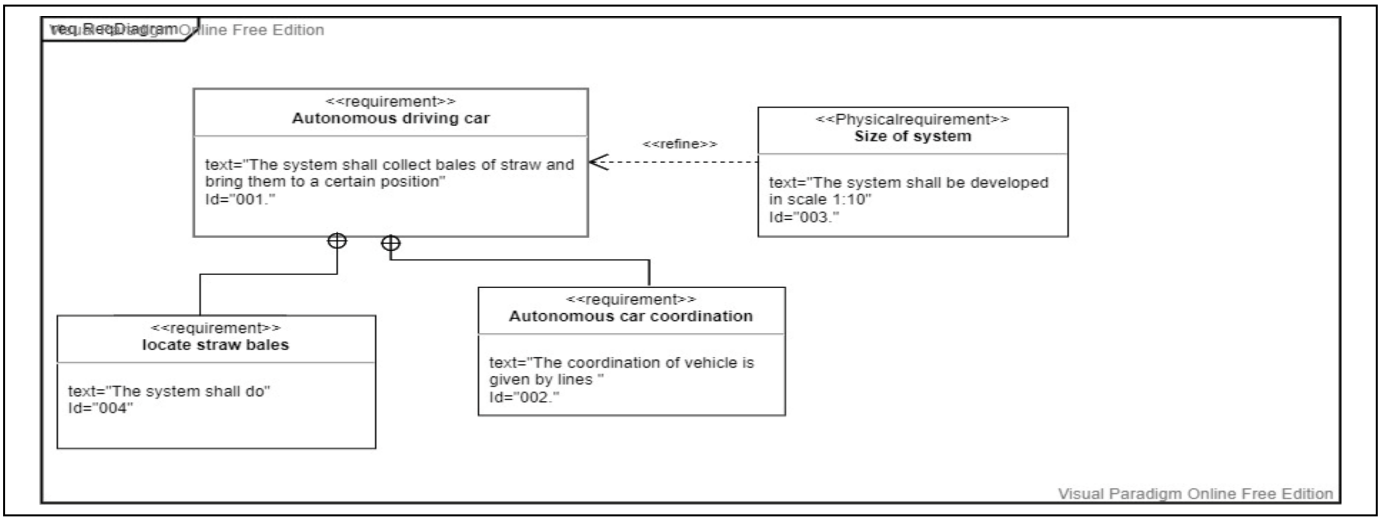

req ReqDiagram
Visual Paradigm Online Free Edition
<<requirement>>
Autonomous driving car
text="The system shall collect bales of straw and bring them to a certain position"
Id="001."
<<refine>>
<<Physicalrequirement>>
Size of system
text="The system shall be developed in scale 1:10"
Id="003."
<<requirement>>
locate straw bales
text="The system shall do"
Id="004"
<<requirement>>
Autonomous car coordination
text="The coordination of vehicle is given by lines "
Id="002."
Visual Paradigm Online Free Edition

Fig 1: requirement diagram first iteration

As displayed below the first iteration of our requirement diagram emphasized the intended functionalities of our system, at the most basic level we highlighted that the smart farming vehicle locate bales, follow the lines and deliver the bales to a certain position, after Professor Dr Henkler reviewed the diagram, he told us to include more details this will be done in future iterations of the design.

b) 2nd iteration of requirement diagram

After input from our professor, we decided to broaden our requirements diagram for the car and added new requirements, the sensors requirement says the system is required to have 2 infra-red sensors and 1 ultrasonic sensor, we added a driving requirement listing all the required components that the system must constitute to drive, an additional turning requirement was added. An image of this diagram is placed in the appendix.

2) State machine

a) The first state machine

Our first state machine (found in the appendix) was corrected by Professor Dr henkler he advised to add a detecting state which we show in the second state machine.

b) ***Second state machine***

In this diagram we have added the detecting state where the vehicle uses the ultrasonic sensors to detect obstacles ahead. (This can also be found in the appendix)

c) ***3rd iteration of the state machine***

Before testing our vehicle, we designed a state machine to help us write the code, this model explains the transitions our system will exhibit from start to finish. It gave an idea of the logic used to accomplish the line follow functionality. (this is also placed in the appendix)

3) *Use case*

a) 1st iteration of Use Case

Our first version of the use case entails a drive vehicle use case with extended functions like collect straw, these functions are conducted by the algorithm, the sensor and actuator. We also have an obstacle detection use case where vehicle needs to recognize lines and colors to help with the car navigation.

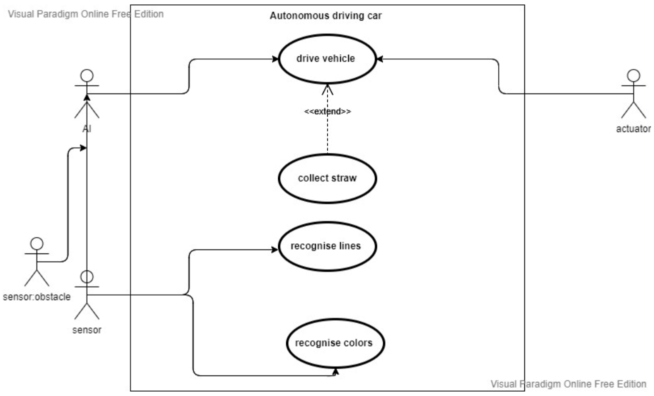


Fig 2: 1st use case

b) Second iteration of use case

In the second iteration we took professor advice and further broaden the drive vehicle use case by adding turn left or turn right as an included functionality we also modeled an obstacle detection use case which Is made possible by the ultrasonic sensor.

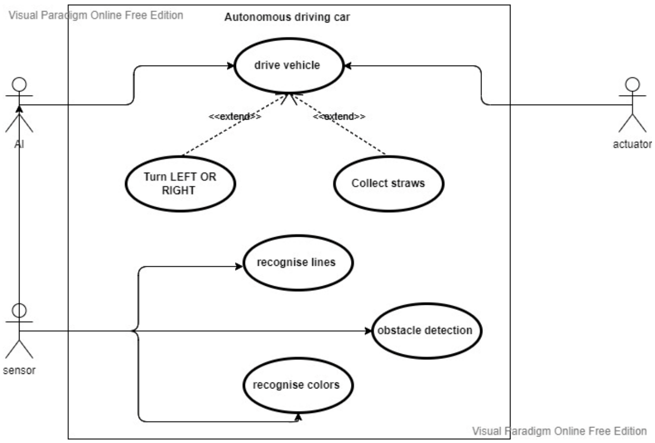


Fig 3:2nd use case

4) *Sequence diagram*

This sequence diagram depicts the sequence of interaction the object of the system will exhibit, sensor scans the environment and sends to the microcontroller , it also scans the size of the bale ,the sensors will also be used to locate the pick up and drop off location the microcontroller controls the picker to pick the bale and generally does the processing of our software the motor is controlled by the microcontroller and the power supply helps to supply power to the vehicle.

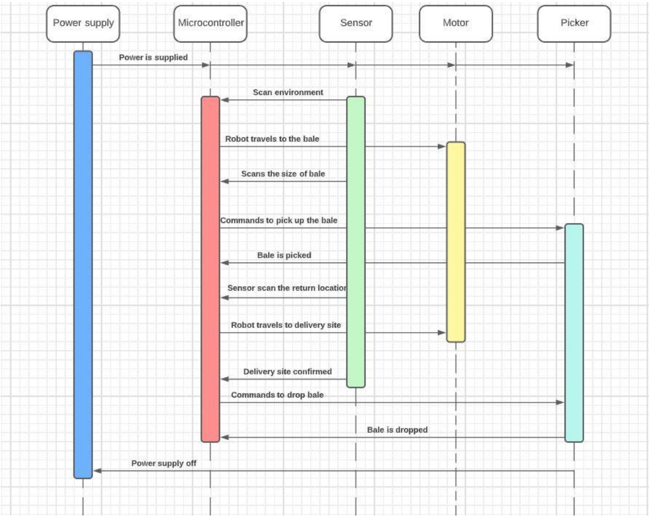


Fig 4 : sequence diagram

5) *Car block diagram*

a) 1st block diagram

This is the block diagram for our car, which the lecturer advised us to expand on in future editions. and we succeeded in doing so.

The diagram entails the operations, parts, references and values of the car giving the viewer a more details description of the vehicle.

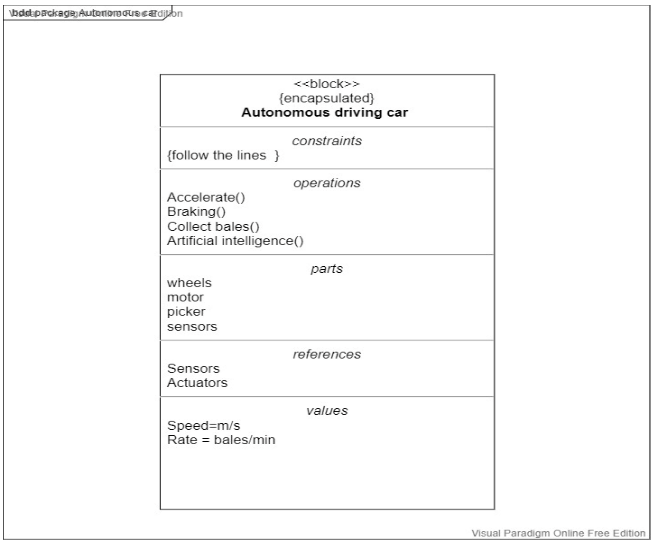


Fig 5: block diagram

b) 2nd block diagram

Expatiated view of system block diagram below

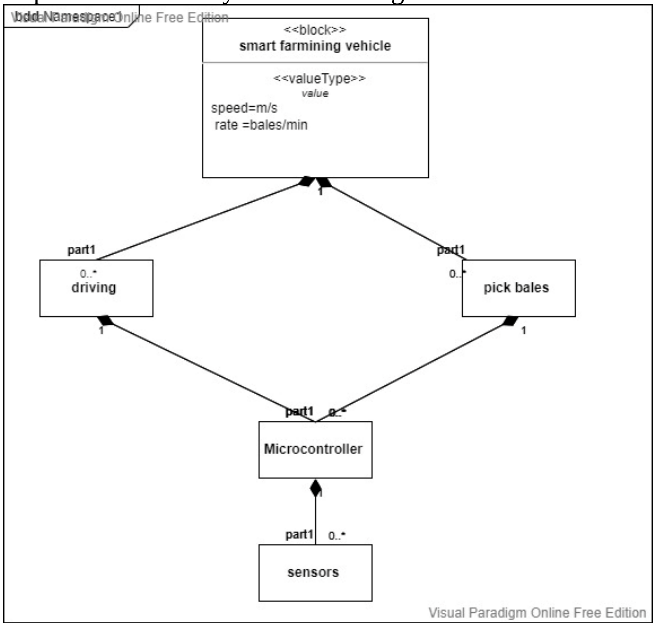


Fig 6:2nd block diagram

6) Constraint diagram

A view of our first constraint diagram.

The major constraint at this stage was the coordination of vehicle by either black or white line.

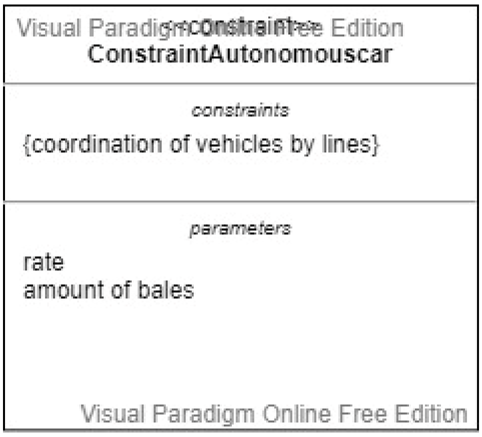

Visual Paradigm Online Free Edition
<<constraint>>
ConstraintAutonomouscar
constraints
{coordination of vehicles by lines}
parameters
rate
amount of bales
Visual Paradigm Online Free Edition

Fig 8: constraint

7) Activity diagram

We modelled the activities the component of the system will embark on from driving to braking to picking up the bales of straw and then to driving to the drop off location.

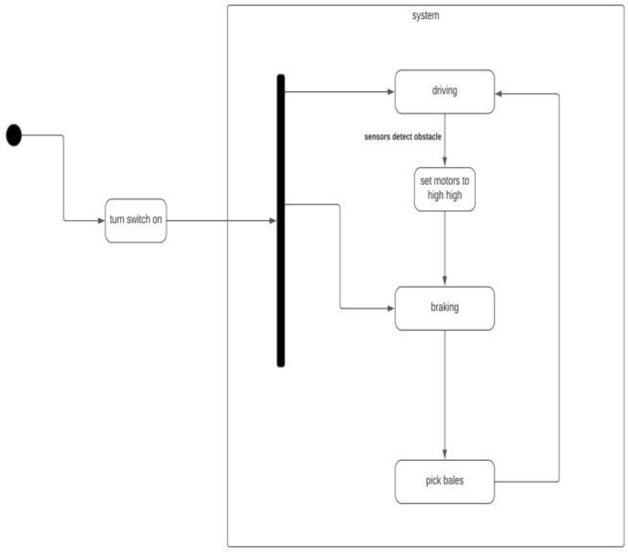


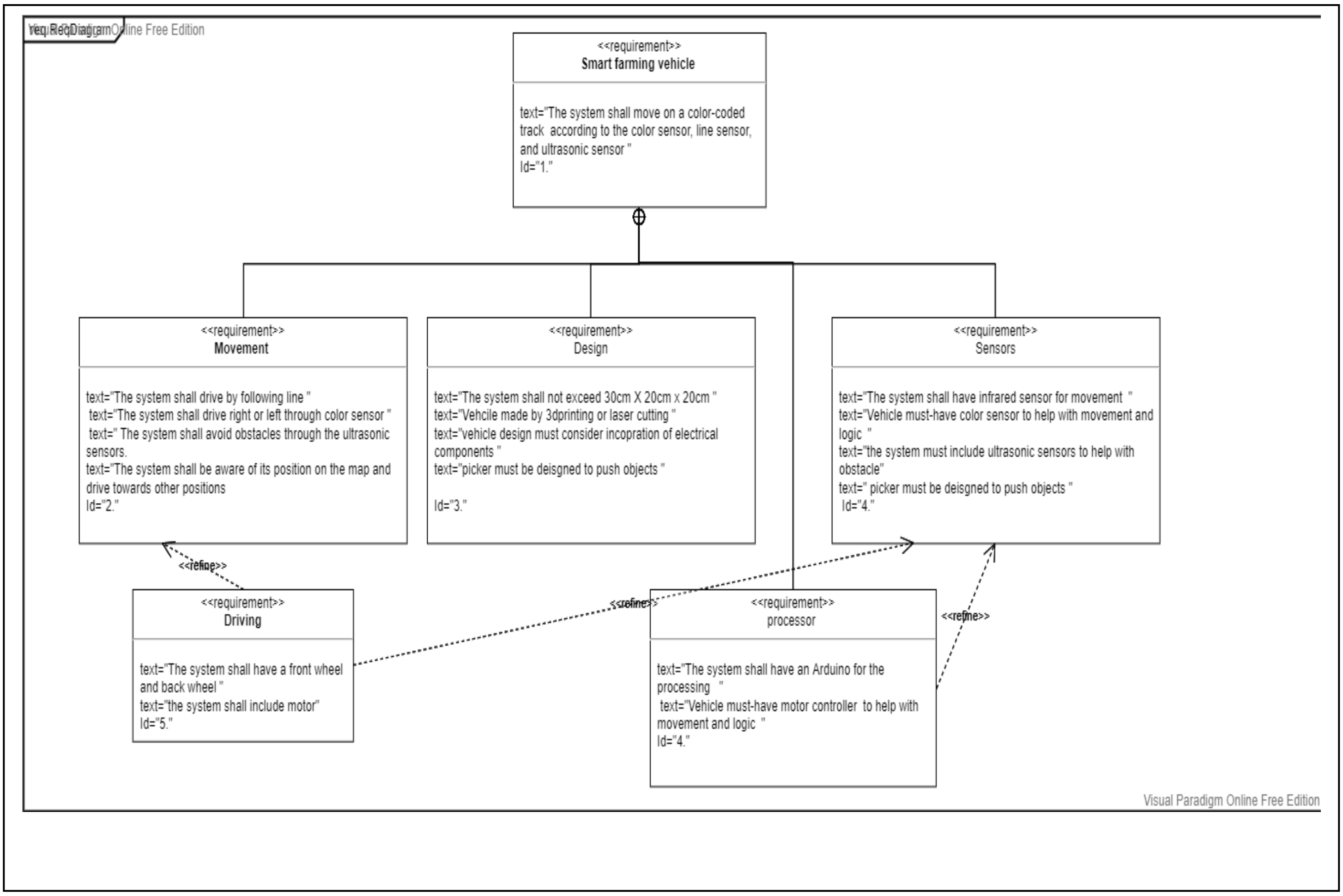
Fig 7: activity diagram

This concludes the system models for the first practical conducted my Professor Dr henkler, we successfully presented our models and showed the movement of the test vehicle

B. *FINAL SYSTEM ENGINEERING MODELS*

We started working on the final project after the first iteration, and the models in this section are iterations of the prior models, better fitting within the enhanced prototyping course requirements. We were given the task of creating a vehicle that runs on color input from lines of various colors. The requirement, transitions and activities of this vehicle will be modeled in this section.

1) Requirement diagram for final Use case


req ReqDiagram
Visual Paradigm Online Free Edition
<<refine>>
<<refine>>
<<refine>>
Visual Paradigm Online Free Edition

<<requirement>>
Smart farming vehicle

text="The system shall move on a color-coded track according to the color sensor, line sensor, and ultrasonic sensor "
Id="1."

<<requirement>>
Movement

text="The system shall drive by following line "
text="The system shall drive right or left through color sensor "
text=" The system shall avoid obstacles through the ultrasonic sensors.
text="The system shall be aware of its position on the map and drive towards other positions
Id="2."

<<requirement>>
Design

text="The system shall not exceed 30cm X 20cm x 20cm "
text="Vehcile made by 3dprinting or laser cutting "
text="vehicle design must consider incopration of electrical components "
text="picker must be deisgned to push objects "

Id="3."

<<requirement>>
Sensors

text="The system shall have infrared sensor for movement "
text="Vehicle must-have color sensor to help with movement and logic "
text="the system must include ultrasonic sensors to help with obstacle"
text=" picker must be deisgned to push objects "
Id="4."

<<requirement>>
Driving

text="The system shall have a front wheel and back wheel "
text="the system shall include motor"
Id="5."

<<requirement>>
processor

text="The system shall have an Arduino for the processing "
text="Vehicle must-have motor controller to help with movement and logic "
Id="4."

FIG REQUIREMENT

For our final requirement diagram, we list out all the requirements that will allow us to achieve the functional and nonfunctional task. For instance,

a) Movement requirement

Our car is required to move from one position to another in the final task. This is done by the motors and wheels as explained in the driving requirement, it is also done by the sensors collecting information, the infra-red the rgb sensor and the ultrasonic, there after sending this information to the Arduino to process. The movement required is following the colored line through line sensor and also following through the rgb sensor, our system must be modeled to achieve this requirement. The system must also avoid obstacles and have the ability to ascertain its own position on the map.

b) Design Requirement

The design requirement tells us about the physical requirements that our vehicles need to exhibit. It tell us the constraints given to the size of our overall vehicle, it also tells us that our vehicle must contain the specified hardware components like breadboard, Arduino motor controller, motor wheels and others .It also tell us what medium can be used to achieve this design , which are 3d printing and laser cutting .Another requirement is that the vehicle must be designed with a picker in the front to push objects as this is required for better functionality of the vehicle.

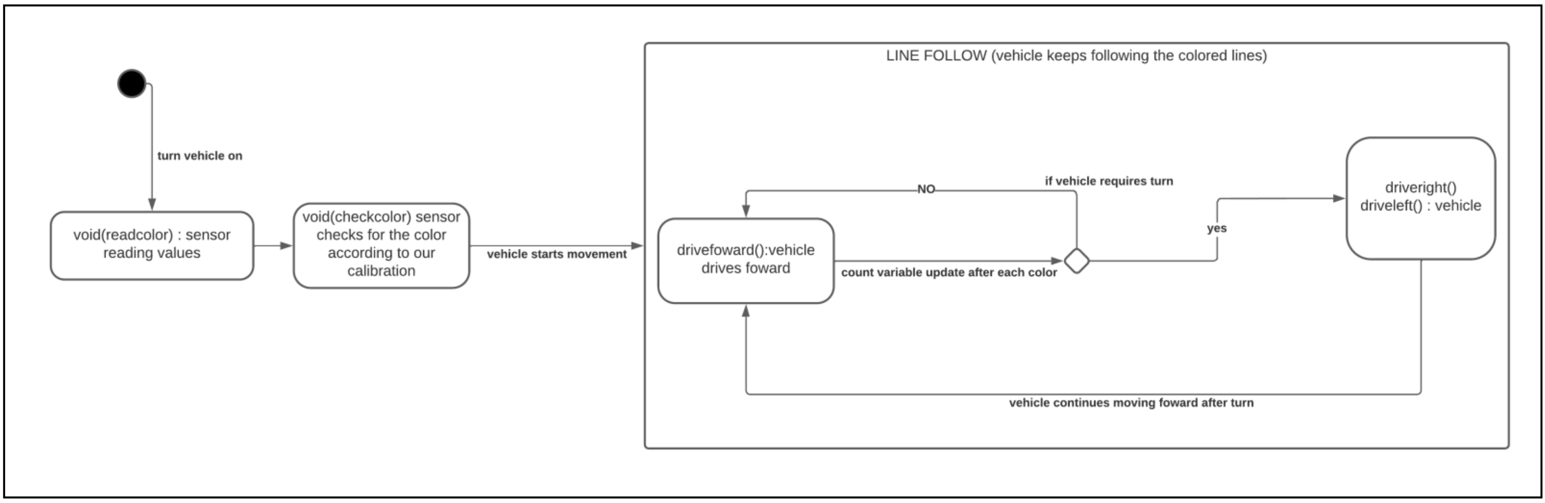
c) Sensors requirement

Our vehicle is required to have 3 different sensors all serving for different purposes. The infra-red sensor for movement and distinguishing between white and black lines, the Rgb sensors for distinguishing color and thereafter helping with the vehicle movement logic. The sensor is connected to the Arduino which processes all this data and performs actins based on the algorithm .

d) Processor requirement

Our intended system must contain an Arduino for processing of the data supplied by the sensors ,the arduino connection is well defined in the caballing image provided. The system must also include a motor controller to serve as an actuator for our vehicle. The motor controller helps coordinate our electric motor.

2) State machine



turn vehicle on
void(readcolor) : sensor reading values
void(checkcolor) sensor checks for the color according to our calibration
vehicle starts movement
LINE FOLLOW (vehicle keeps following the colored lines)
drivefoward():vehicle drives foward
NO
if vehicle requires turn
yes
count variable update after each color
driveright() driveleft() : vehicle
vehicle continues moving foward after turn

Fig 10: state machine

The above state machine explains the transitions and different states our system will exhibit. After turning on the car the void(readcolor;) function is activated in this function we read the values generated from the Rgb sensor and store them in a variable named pulse width ,after the values are store we activate the void(checkcolor) function where we check to see the color being read by the sensor out of the colors we already calibrated(green orange purple and blue).After checking the color the drive(forward) function is activated if the color checked meets the requirement of our algorithm. While the vehicle is driving it continuously runs the check color state and stores every color in a count variable, the turning of our vehicle is based on this variable when the required count value has been reached the car transitions into the drive right or drive left state dependent on the position the car is going to, after this the car transitions back to the drive forward state and till it arrives its destination. The overall idea is for the car to transition from the sensor and processing state to the action states like driving forward and turning right or left. During the drive states the car would also be in the line follow state, which means that the vehicle will always follow the colored line and adjust itself appropriately.

3) Use case diagram

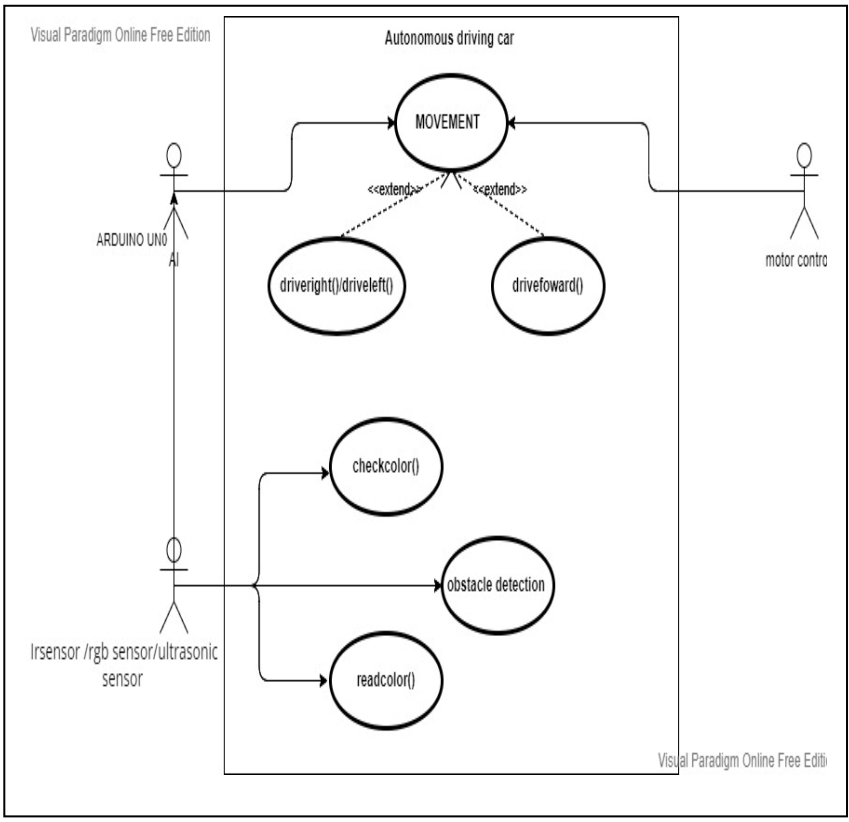


Fig 11: use case

The diagram above depicts the use case,giving us an idea of the various functionalities of our system. The movement case contains 2 other use cases, the turning and the forward drive, these are the most obvious functionalities in our system, but underneath the hood we have the sensors and their use cases, the IRsensors and rgb sensors are used majorly for the coordination of our cars with the motor controller, Arduino and our software being the actors in this system.

4) *Constraints*

Our vehicle will be constraint by the lines and the color of the lines. The IRsensor will look for colored lines (high, high) and ensure that the vehicle continues on this lines. The color sensors will give us input to be aware of the vehicle current placement on the map and also to know when to turn. The count variable holds this input and helps us to constraint the system in a particular manner. Flag is set to 1 every time we sense a new pair of colors this also helps us while turning.

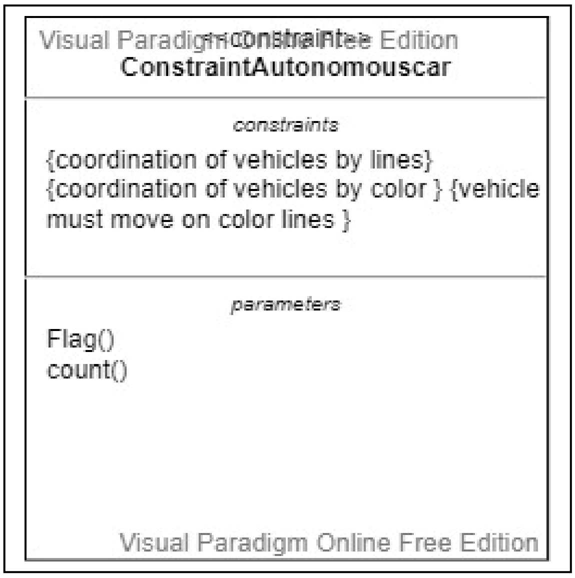
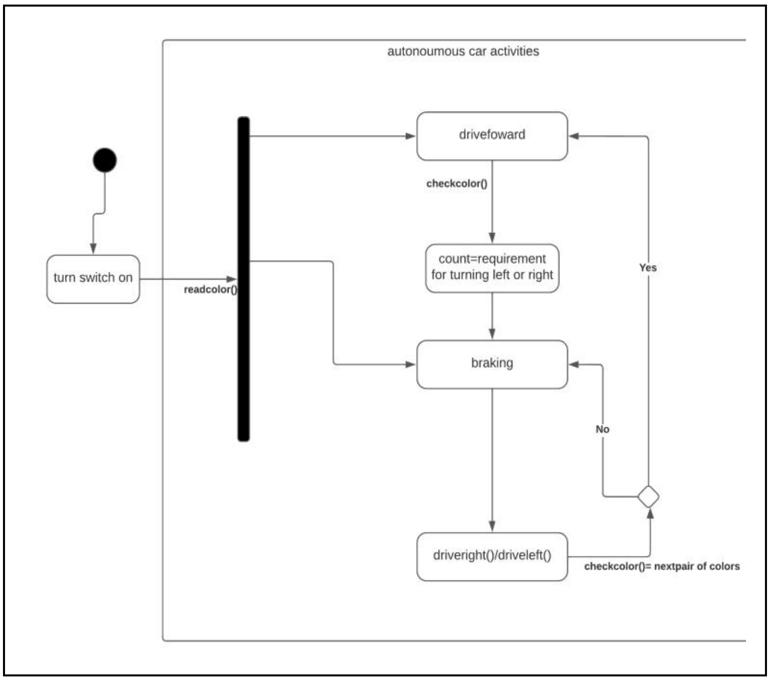

Visual Paradigm Online Free Edition
«constraint»
ConstraintAutonomouscar
constraints
{coordination of vehicles by lines}
{coordination of vehicles by color } {vehicle must move on color lines }
parameters
Flag()
count()
Visual Paradigm Online Free Edition

Fig 12 : constraint

5) *Activity*


autonoumous car activities
turn switch on
readcolor()
drivefoward
checkcolor()
count=requirement for turning left or right
braking
driveright()/driveleft()
checkcolor()= nextpair of colors
Yes
No

In the activity diagram we see a representation of the flow of activities in our system. This diagram depicts the system behavior in moving from the start coordinate [0,0] to any defined coordinate. The track consist of the horizontal and vertical position which have unique color combination. with the help of the rgb sensor we will be able to differentiate between these colors and store them in a variable called count. The system when turned on starts in a state of reading color values after this we check for the colors after checking the colors the system starts driving forward when count is equal to the required amount our system will detect it and realize that it is time to make a turn, the system will brake and slowly turn to align itself with the new set of colors. The system will continue breaking and turning until the checkcolor() function gives a new set of values. After this is done our system resumes drivefoward() till it reaches the intended destination.

6) *Block definition diagram*

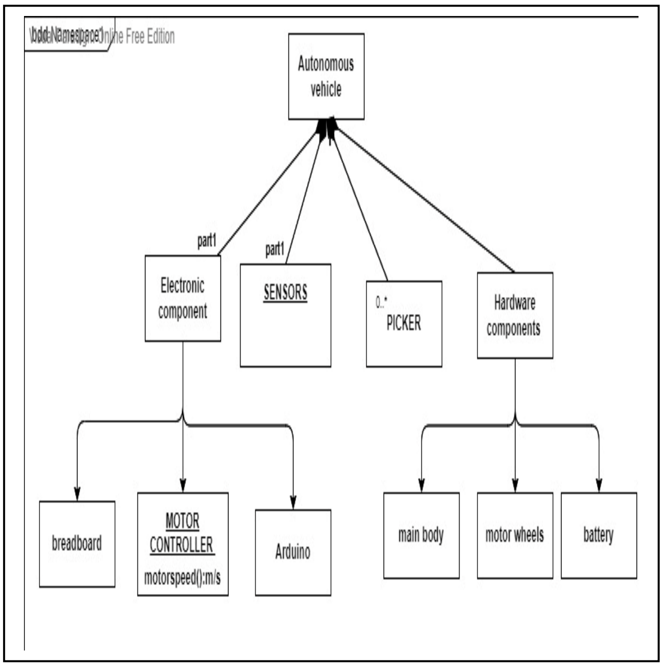


Fig 14 : BBD

The above diagram shows us the system we are designing in detail; it exhibits the relationship between each system components and it shows the content. This is a structural view of the system. The system is split into the main body, the electronic components, the picker and the hardware components. The main body is where we house the battery the wheels and the other electronic components like Arduino and breadboard, the picker is a part of the vehicle that is attached onto the vehicle, the sensors are placed at the front side of the vehicle and carry out very important task, we also have the electronic components like the motor controller, Arduino and breadboard. All together this forms the system that Is our autonomous vehicle.

7) Sequence diagram

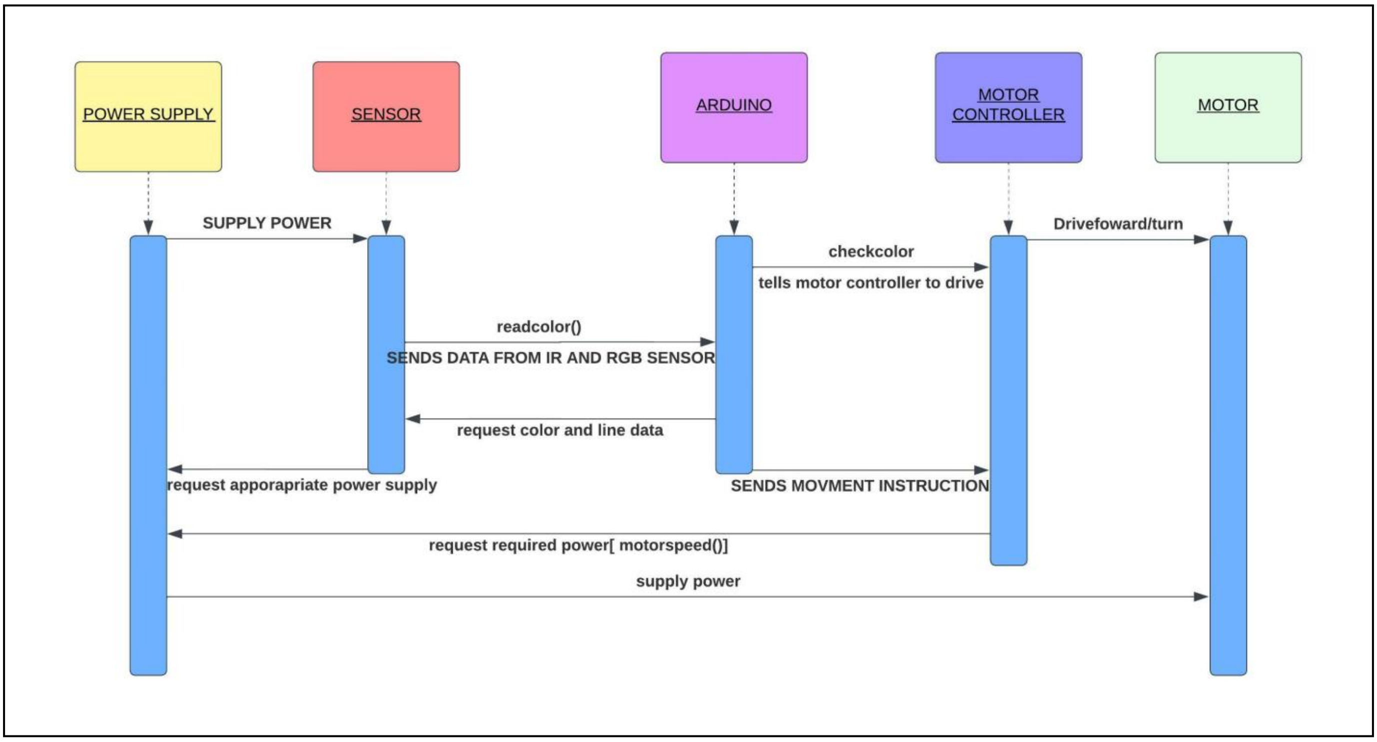

POWER SUPPLY
SENSOR
ARDUINO
MOTOR CONTROLLER
MOTOR
SUPPLY POWER
Drivefoward/turn
checkcolor
tells motor controller to drive
readcolor()
SENDS DATA FROM IR AND RGB SENSOR
request color and line data
request apporapriate power supply
SENDS MOVMENT INSTRUCTION
request required power[motorspeed()]
supply power

Fig 15 : sequence diagram

The sequence diagram above shows the series of interactions between the components of our system. When the car is turned on, the power supply is activated and it supplies the required power requested by the system components as the sensors and micro controller we used have different levels of power output, after this the Ir sensor and rgb sensor sends the data of their readings to the Arduino using the readcolor() function the Arduino requests for this data as it will be used in the defined algorithm. The Arduino interacts with the motor controller by sending commands for the motor controller to carry out, this could be brake, drivefoward(), driveright() or driveleft(). The motor controller in turn controls the motor in the appropriate manner. The speed of the motor is dependent on the motor controller which is in turn dependent on the power supply. These are the most important interactions or system will exhibit.

III. DESIGN

In the design phase, we were given the responsibility of designing and constructing our own vehicle. We had two alternatives for creating our design: laser cutting or 3D printing. Our group employed a hybrid of the two, with a heavier emphasis on 3D printing. The following is a step-by-step breakdown of our design process, including design iterations, component assembly, and final output.

A. *Initial idea and designs*

We had a solid notion of how our vehicle should look after attending the first lab practical and presentation with Professor Dr. Henkler, the only new component necessary was the picker, which was needed to push the bales to a desired destination. We devised a design for this and refined it over time to make it more functional. Our design was concentrated on the hardware component and how it integrated into the overall design. We took into account factors like battery weight, breadboard length, and other design concepts. The following is a list of the original concepts and sketches that we came up with.

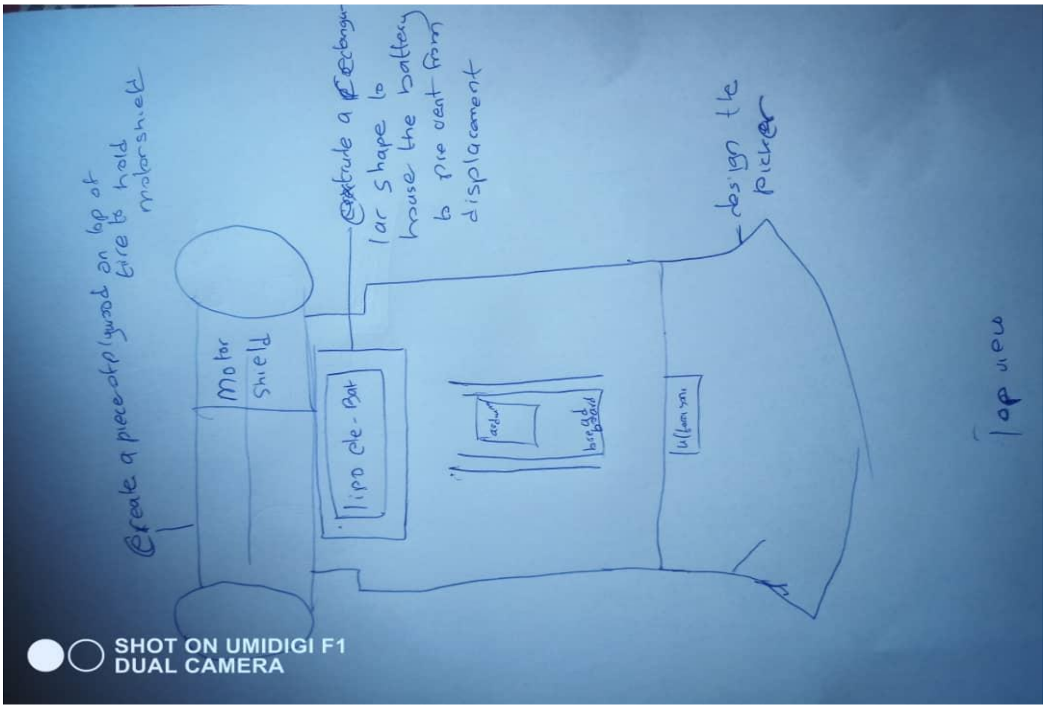

Create a piece of plywood on top of tire to hold motorshield
Motor shield
lipo cle - Bat
extrude a rectangular shape to house the battery to prevent from displacement
arduino
bread board
ultrasonic
design the picker
Top view
SHOT ON UMIDIGI F1
DUAL CAMERA

Fig 16: initial sketch

1) *Initial sketch*

Above in fig 10 was our initial sketch idea, to put the breadboard in the middle along with the battery but when we tried this we found out it exceeded the required dimension on all axis , so we started a new design taking the dimensional restriction as a very important part of the design.

2) *Second iteration*

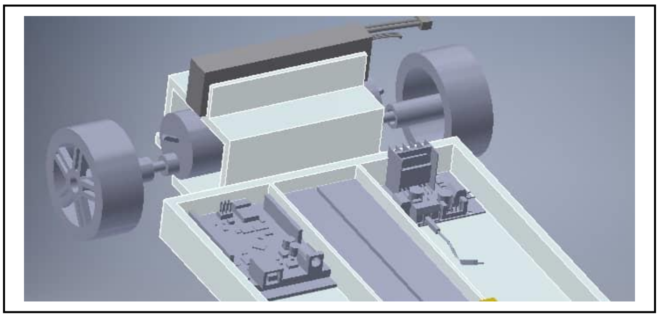


Fig 17: second iteration

For the second iteration we decided to put the battery on top of the motors safeguarding the motors in a compartment below and then put the breadboard in the main body of the vehicle, however this created another problem as the battery position and weight could affect the car movement, we needed to now consider aligning the batter weight with the center of mass of the vehicle.

3) *Third iteration*

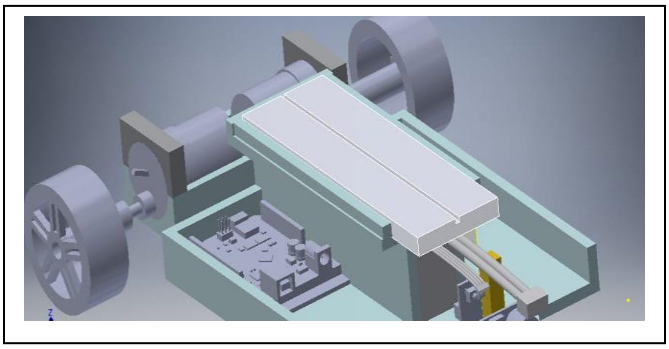


Fig 18: 3rd iteration

In our 3rd design we decided to create a compartment to hold the battery in the middle of the main body helping us spread the weight of the battery along the length of the vehicle. We also put the Arduino and motor controller right beside the vehicle and the breadboard on top. Fig 12 also shows the clamps we designed to hold down the motors.

4) *Fourth iteration*

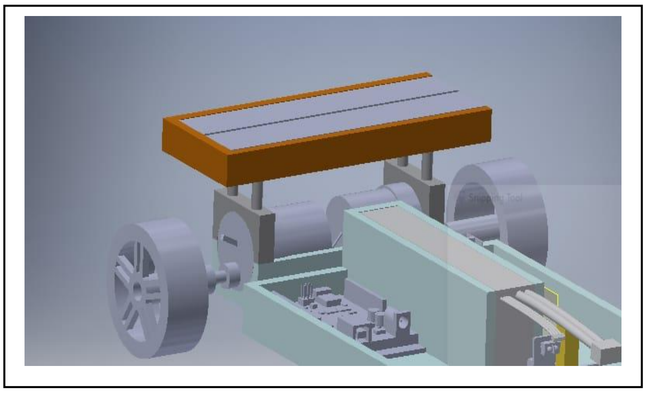


Fig 19: 4th iteration

In the fourth design we created a breadboard stand to contain the breadboard also using the stand to clamp down on the motors at this stage our final design was already coming alive we however still had the picker to design.

5) *Picker design*

a) First iteration

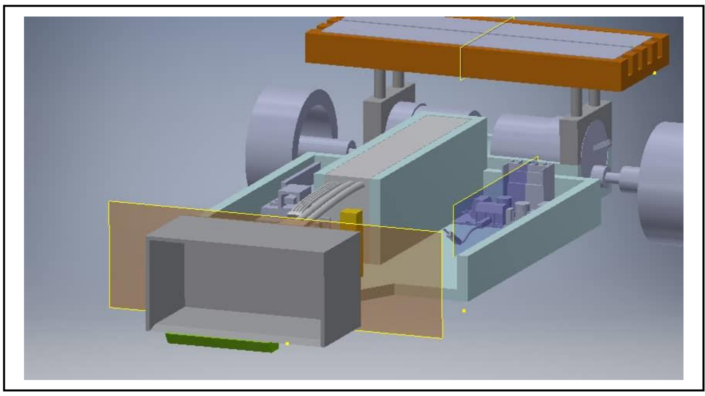


Fig 20: 1st picker design

Above was our initial idea for a picker after discussing withing the team and studying the mechanics of the system we decided that this design might not prove sufficient as the boxes are not well contained. we decided to give the vehicle arms like a tractor instead of just the box above.

b) Second picker iteration

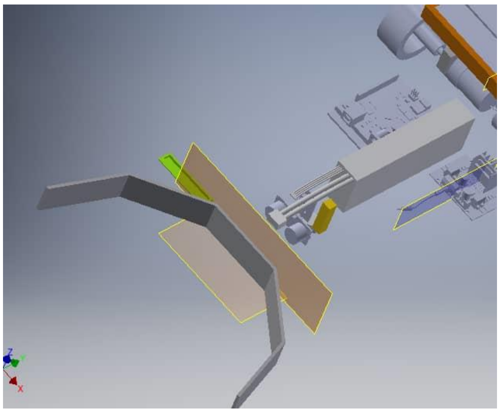


Fig 21 : 2nd picker design

Above is the picker we redesigned , enabling the vehicle with arms so as to hold the vehicle even better whilst turning, at this stage our design work was almost complete and we were ready for printing our design.

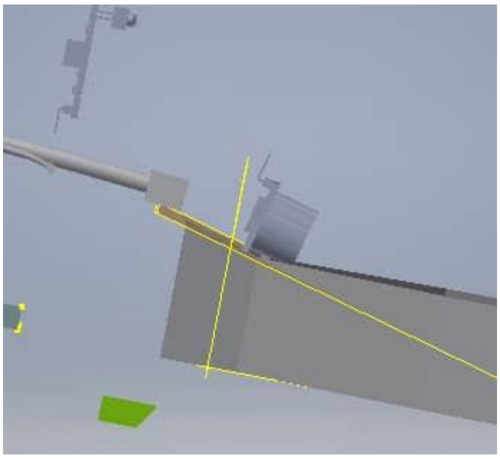


fig 22: ultrasonic sensor concept

we had a concept for making our ultrasonic sensor tilt downwards we designed a hole on top of the picker so when place it bends downwards this concept was later scrapped as we found a better way to place the sensor.

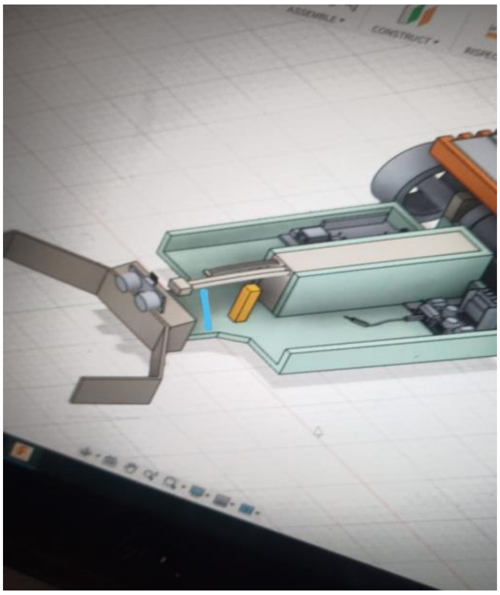


Fig 23: better view of design

c) Picker and body conncetion

For the picker and body connection we decided to join them by creating holes in the picker which extending part of the body will enter. This was the concept we use to satisfy the problem.

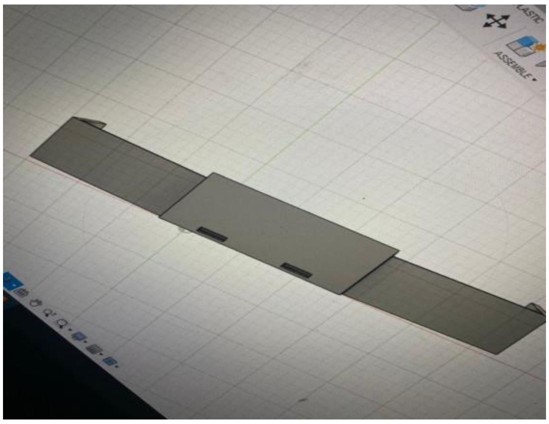


Fig 24: holes in picker

B. Printing assembly and additional design

In this section, we explain the process challenges and innovations we encountered during this process. 3D printing was a bit time consuming and had a lot of requirements that we had to strictly adhere to, in order for our object to print. After finishing the design, we used the Felix printer structure to test our vehicle design to see if it was beyond the dimension, which it was, but since the professor said we could print as components, we divided our vehicle into three parts, the motor the main body and the picker. But even after the division our main body was still above the requirement causing us to cut out some length in the front as shown below.

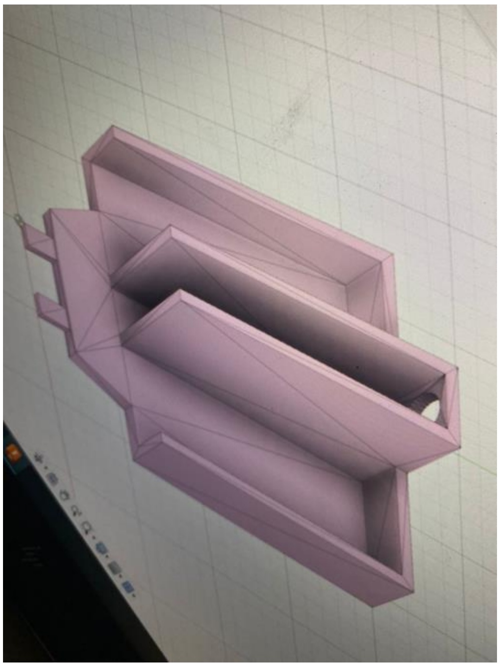


Fig 25: updated body for printing

1) 3d printing

After our design fulfilled all requirement for printing, we submitted the STL files to the professor to be printed we were told to return the next week.
After collecting the 3d printed components of our vehicle (main body, motor compartment and picker) we moved on to the assembly.

2) Assembly

After printing the components our next task was assembling all components together and attaching the electrical components on our vehicle. Like we mentioned earlier there are 3 major components printed separately because they exceeded the required dimension for the Felix printers. We started with the motor compartment.

a) Motor compartment

Our motor compartment consisted of the motors, the motor clamps, the breadboard stand and the breadboard holder. Our task was to tightly and firmly fit all these components together. We started by making sure the motor was tightly contained at the axis of the vehicle. We laser cut strips of wood to use to make sure our motor was tightly fitted after this we proceeded to attach our breadboard to the stand but discovered it couldn't contain this caused us to finally decide to turn it upside down and attach the breadboard by means of rubber band as shown in fig 20

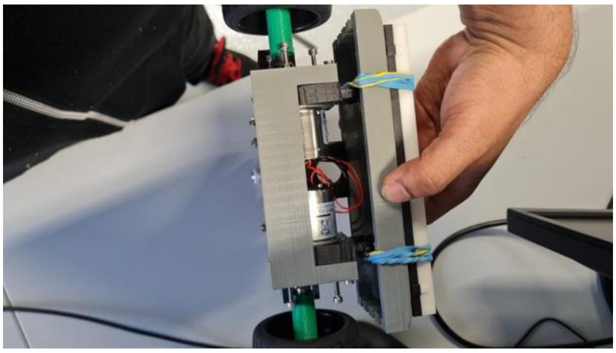


Fig 26:breadboard

After fixing the breadboard stand and the motor clamps in the motor compartment we proceeded to the main body.

3) Main body

The main body consists of the battery compartment the Arduino and the motor controller. Our battery fit perfectly into the batter compartment we designed. The motor controller was also attached by drill and screws the Arduino Was attached in the same manner.

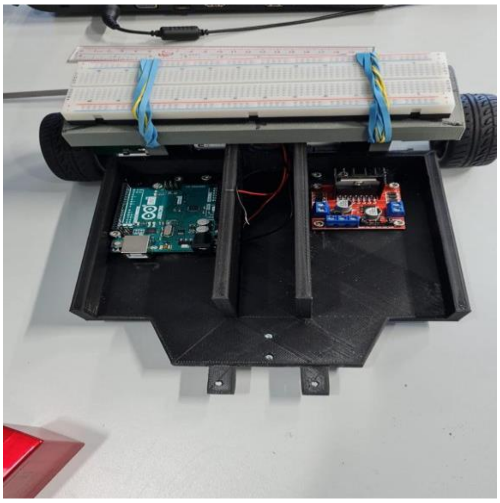


fig 27:main body

4) Sensors attachment

For the sensors we attempted to attach them to the car directly but later decided against it due to the functional requirement, our new idea to attach the sensors was creating a platform where the Infra-red sensors and the color sensors will both be attached we designed a platform measuring 10cm by 11cm on this platform the color sensor and infra-red sensor were placed and the ends of the sensor was attached to both the picker and body. Holes for the passage of wires were also included in our structure.

The ultrasonic sensor was however installed at the top of our picker by laser cutting a predesigned box that will hold the sensor at an angle.

C. Wiring

The wiring of our vehicle was done according to the specified cable Image. we already made holes in the vehicle to help with the passing of cables from one position to another, a picture of our vehicle after cable completion can be seen below.

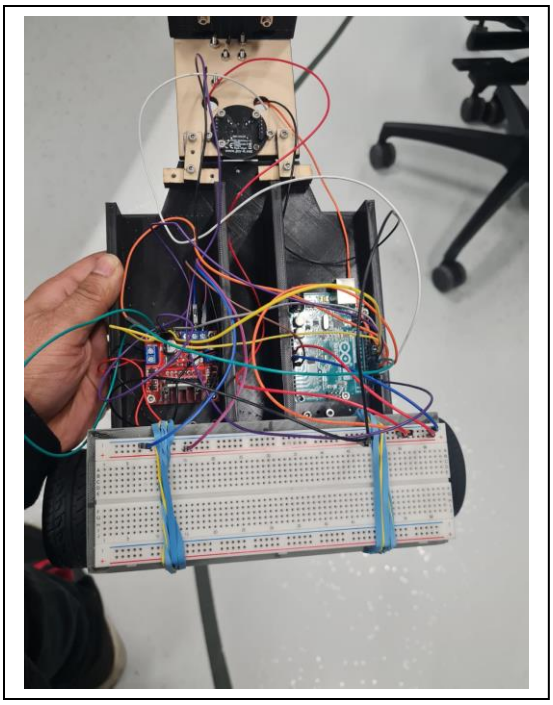


Fig 28: vehicle after wiring

DESIGN CONCLUSION

The whole design process was immensely interesting and fulfilling we went through the stages of design to bring to life a Vehicle that is capable of movement and pushing of balls. We experienced the iterative nature of designing ,we had to go back to our design software to redesign and correct a lot of times in the first stage of printing, we also had to design innovative structures to help with the functional requirement of our cars such the motor clamps holding the motor tightly down , the breadboard stand and the platform for the 2 sensors. The overall experience was challenging but also worthwhile we applied various techniques we acquired in previous semesters about design process and came up with a very unique vehicle. We were able to complete this part of our project due to hard work and most especially teamwork.

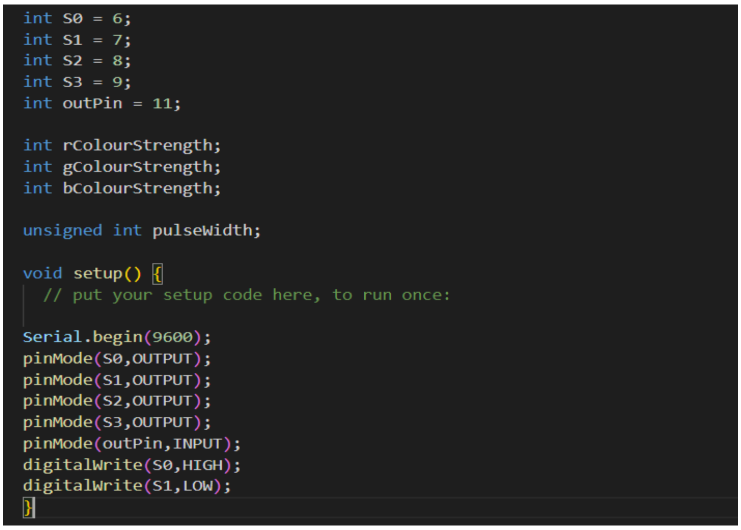
IV. PROGRAMMING

After completion of the systems models and the design part of our system, our next task was programming the vehicle to perform certain functions listed out by our professor. we used both the IR sensor and the rgb sensor to accomplish this task.

1) *Calibration*

Our first task was calibrating the rgb sensors this proved a little difficult as we experienced a recurring bug in our code, but alas we were able to get it right.

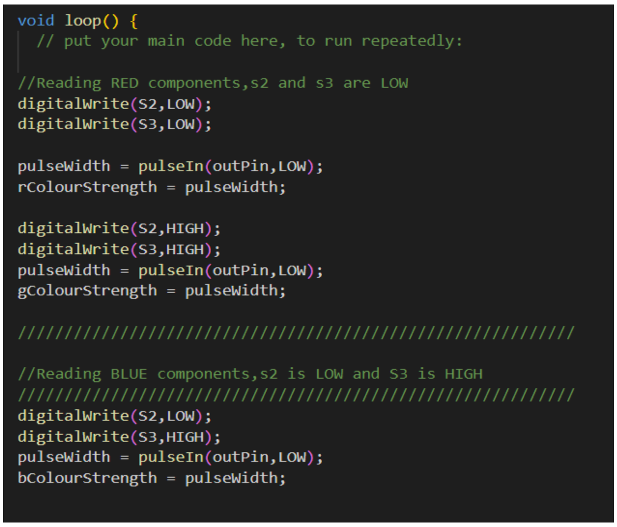
Our code for this part includes the standard declarations and then a pulse width function where the values of the sensors are stored.



```
int S0 = 6;
int S1 = 7;
int S2 = 8;
int S3 = 9;
int outPin = 11;

int rColourStrength;
int gColourStrength;
int bColourStrength;

unsigned int pulseWidth;

void setup() {
  // put your setup code here, to run once:

Serial.begin(9600);
pinMode(S0,OUTPUT);
pinMode(S1,OUTPUT);
pinMode(S2,OUTPUT);
pinMode(S3,OUTPUT);
pinMode(outPin,INPUT);
digitalWrite(S0,HIGH);
digitalWrite(S1,LOW);
}
```

Fig : setup code

Below is our main loop that include the code we used to collect the data from the sensor pins, alternation of the low and high parameters will gives us values for different colors.



```
void loop() {
  // put your main code here, to run repeatedly:

//Reading RED components,s2 and s3 are LOW
digitalWrite(S2,LOW);
digitalWrite(S3,LOW);

pulseWidth = pulseIn(outPin,LOW);
rColourStrength = pulseWidth;

digitalWrite(S2,HIGH);
digitalWrite(S3,HIGH);
pulseWidth = pulseIn(outPin,LOW);
gColourStrength = pulseWidth;

//////////////////////////////////////////////////////////

//Reading BLUE components,s2 is LOW and S3 is HIGH
//////////////////////////////////////////////////////////
digitalWrite(S2,LOW);
digitalWrite(S3,HIGH);
pulseWidth = pulseIn(outPin,LOW);
bColourStrength = pulseWidth;
```

Fig: main loop

After this a serial.println() function is used to print the results of this to the serial monitor.

2) *Functions*

After our sensors were calibrated, we began with some simple functions that will be listed below.

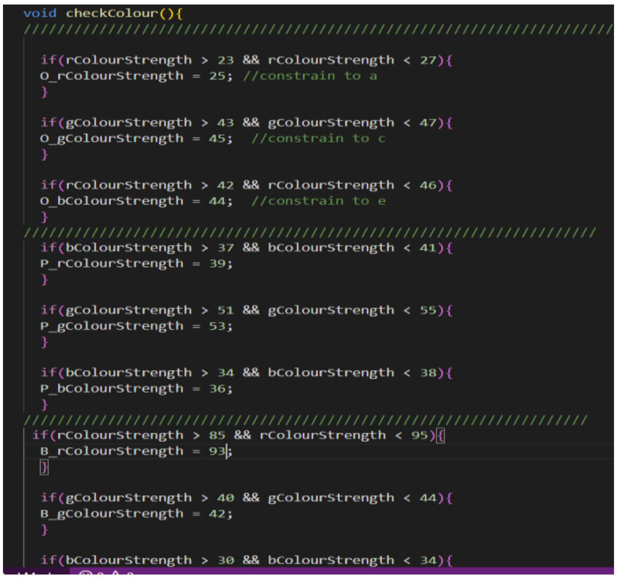
a) *Readcolour()*

The function read color is the same as our void loop used for calibration it reads, the RGB values of different colors and stores them in rcolorstrenght, bcolorstrenth, gcolorstrenght variables as shown in fig: main loop.

b) *Checkcolour()*

After reading the color we came up with a function that verifies the actual color we are sensing based on the formerly stored color values. This consist of 2 parts.

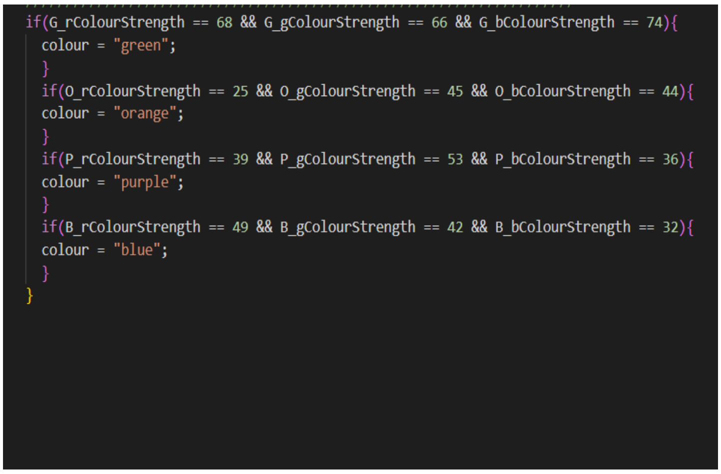
The first part is a range set algorithm in which we set a range of values which ultimately results in one value, this was done because the color sensor might not deliver a constant value consistently, so we need a means to fine tune the results by adding a higher range of acceptable values below is the code.



```
void checkColour(){
//////////////////////////////////////////////////////////////////////

  if(rColourStrength > 23 && rColourStrength < 27){
  O_rColourStrength = 25; //constrain to a
  }

  if(gColourStrength > 43 && gColourStrength < 47){
  O_gColourStrength = 45;  //constrain to c
  }

  if(rColourStrength > 42 && rColourStrength < 46){
  O_bColourStrength = 44;  //constrain to e
  }
/////////////////////////////////////////////////////////////////////
  if(bColourStrength > 37 && bColourStrength < 41){
  P_rColourStrength = 39;
  }

  if(gColourStrength > 51 && gColourStrength < 55){
  P_gColourStrength = 53;
  }

  if(bColourStrength > 34 && bColourStrength < 38){
  P_bColourStrength = 36;
  }
////////////////////////////////////////////////////////////////////
 if(rColourStrength > 85 && rColourStrength < 95){
  B_rColourStrength = 93;
  }

  if(gColourStrength > 40 && gColourStrength < 44){
  B_gColourStrength = 42;
  }

  if(bColourStrength > 30 && bColourStrength < 34){
```

Fig: range set

The second part is a simple series of if() statements which is responsible for setting our colour variable to the read values.



```
if(G_rColourStrength == 68 && G_gColourStrength == 66 && G_bColourStrength == 74){
  colour = "green";
  }
  if(O_rColourStrength == 25 && O_gColourStrength == 45 && O_bColourStrength == 44){
  colour = "orange";
  }
  if(P_rColourStrength == 39 && P_gColourStrength == 53 && P_bColourStrength == 36){
  colour = "purple";
  }
  if(B_rColourStrength == 49 && B_gColourStrength == 42 && B_bColourStrength == 32){
  colour = "blue";
  }
}
```

3) *Programming task*

A series of task were given to us, detailed in this section is the process in which we applied to achieve these tasks.

a) Drive on the line(vertically and horizintally)

For this use case we implemented a simple line follow functionality in which the car uses the Ir sensor to follow the colored lines, this works both horizontally and vertically, it is similar to the first line follow we implemented but instead of black we follow any line that is not black and constraint our car by use of tilting function to adjust back to the colored lines anytime it senses black, below is a state machine exhibiting above explained scenario.

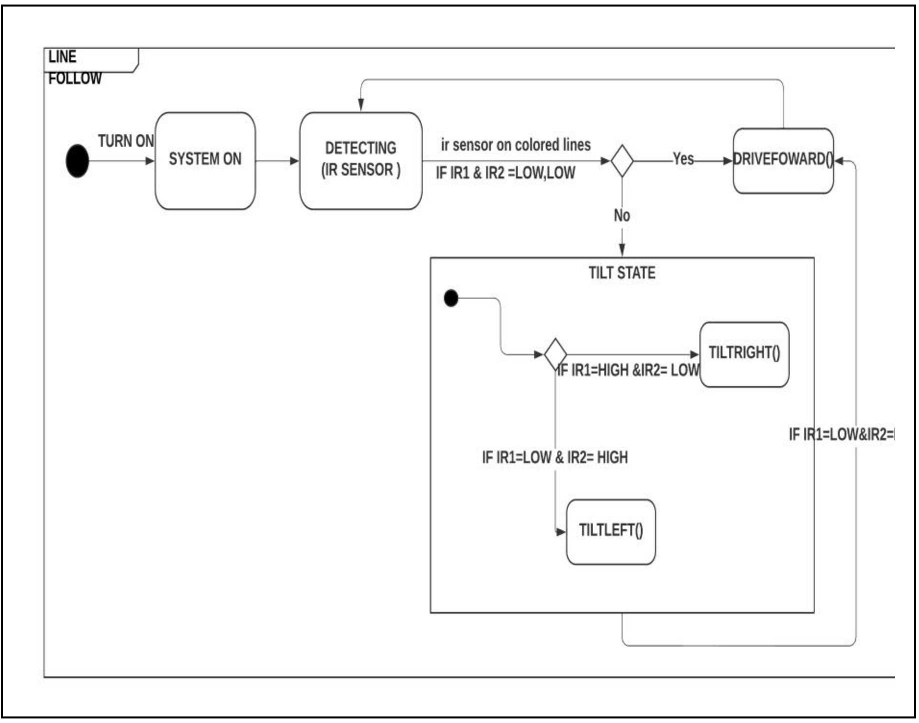
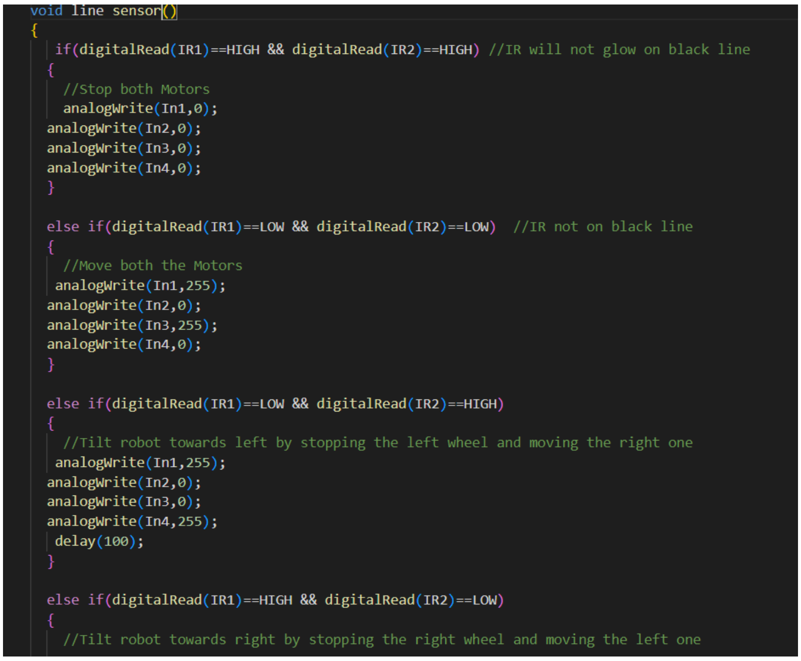


Fig: line follow

Below is a look at the code.



```
void line sensor()
{
  if(digitalRead(IR1)==HIGH && digitalRead(IR2)==HIGH) //IR will not glow on black line
 {
   //Stop both Motors
   analogWrite(In1,0);
 analogWrite(In2,0);
 analogWrite(In3,0);
 analogWrite(In4,0);
 }

 else if(digitalRead(IR1)==LOW && digitalRead(IR2)==LOW)  //IR not on black line
 {
   //Move both the Motors
  analogWrite(In1,255);
 analogWrite(In2,0);
 analogWrite(In3,255);
 analogWrite(In4,0);
 }

 else if(digitalRead(IR1)==LOW && digitalRead(IR2)==HIGH)
 {
   //Tilt robot towards left by stopping the left wheel and moving the right one
  analogWrite(In1,255);
 analogWrite(In2,0);
 analogWrite(In3,0);
 analogWrite(In4,255);
  delay(100);
 }

 else if(digitalRead(IR1)==HIGH && digitalRead(IR2)==LOW)
 {
   //Tilt robot towards right by stopping the right wheel and moving the left one
```

b) Differentiate bewteen all colors on the track

This software requirement is certified by the check color function we have previously discussed in 2b of this section. We have an algorithm in place to read the values of the color the sensor is on then match it with the set of values already stated that represents the blue, orange, purple and green color.

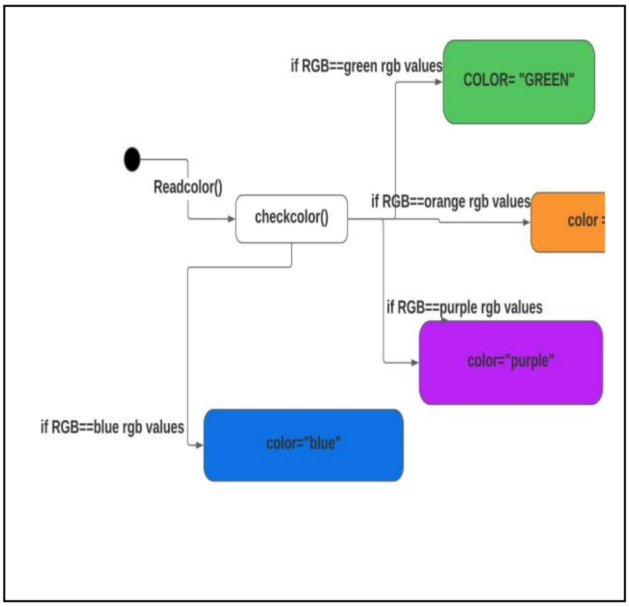


Fig : differentiate color

c) Drive and count (stop car after N intersection)

For the following task we introduced new variables like the countx,county and intersection to help keep count of the colors being red. Our approach to solving this includes initializing intersection variable as zero and increasing it by 1 any time a particular color is read. After it reaches the required amount of intersection, we stop the vehicle and serial. Print the vehicles position on the track.

Below is the state machine and code used.

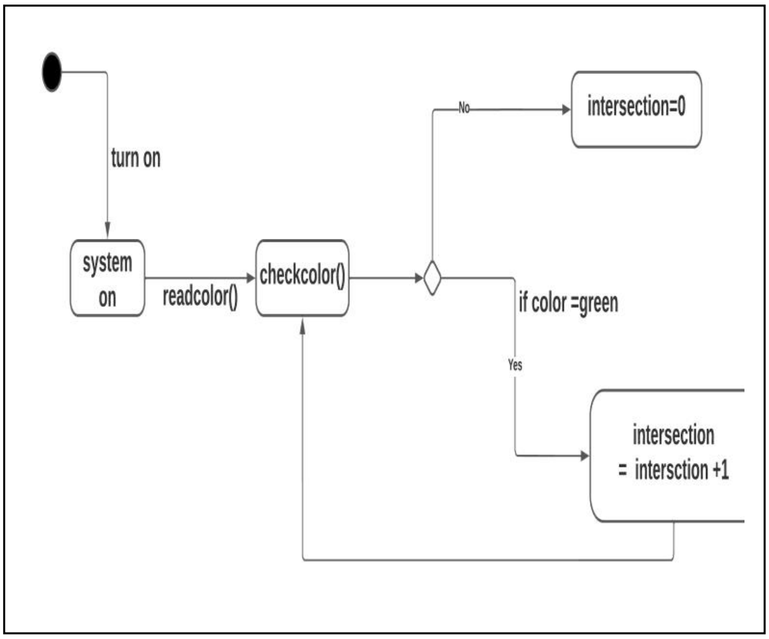


fig: state machine for task 3

4) Drive on the line turn 90 degrees and keep driving

For this task the general idea was to do a line follow logic for the car and some new variables like the countx and flag. The general idea was to readcolor() and check color at first and if this color was = to a color on the vertical axis the car will move , the car continues moving and brakes when it is about to take a turn , at this point it gradually keeps turning until the color sensor reads a different set of colors (colors on horizontal axis) ,when it does our variable flag is set to 1 and the vehicle will enter the move vehicle condition where it will drive forward .

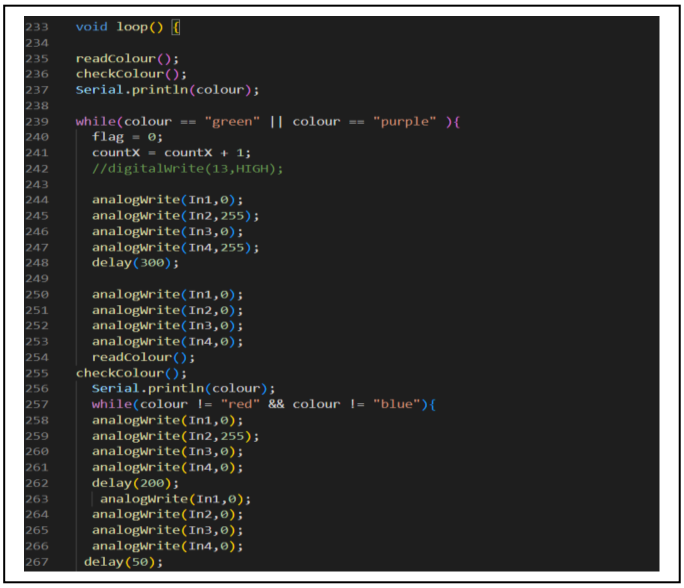

```
void loop() {

readColour();
checkColour();
Serial.println(colour);

while(colour == "green" || colour == "purple" ){
  flag = 0;
  countX = countX + 1;
  //digitalWrite(13,HIGH);

  analogWrite(In1,0);
  analogWrite(In2,255);
  analogWrite(In3,0);
  analogWrite(In4,255);
  delay(300);

  analogWrite(In1,0);
  analogWrite(In2,0);
  analogWrite(In3,0);
  analogWrite(In4,0);
  readColour();
checkColour();
  Serial.println(colour);
  while(colour != "red" && colour != "blue"){
  analogWrite(In1,0);
  analogWrite(In2,255);
  analogWrite(In3,0);
  analogWrite(In4,0);
  delay(200);
   analogWrite(In1,0);
  analogWrite(In2,0);
  analogWrite(In3,0);
  analogWrite(In4,0);
 delay(50);
```

Fig :first part of code for task 4

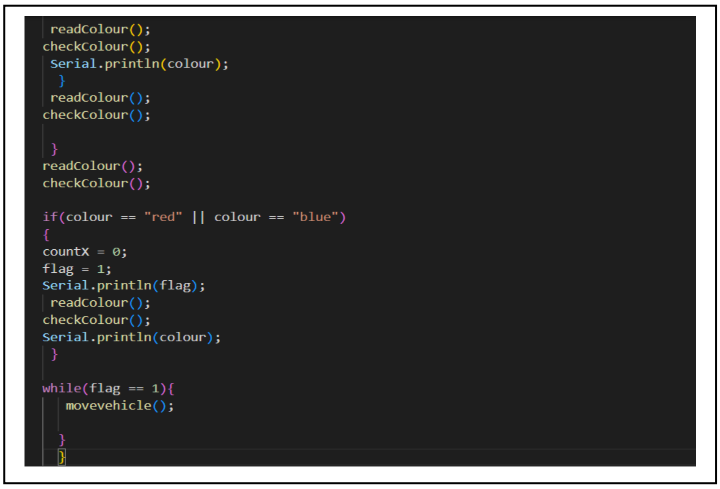

```
 readColour();
checkColour();
 Serial.println(colour);
  }
 readColour();
checkColour();

 }
readColour();
checkColour();

if(colour == "red" || colour == "blue")
{
countX = 0;
flag = 1;
Serial.println(flag);
 readColour();
checkColour();
Serial.println(colour);
 }

while(flag == 1){
   movevehicle();

  }
  }
```

Fig : second part of task 4

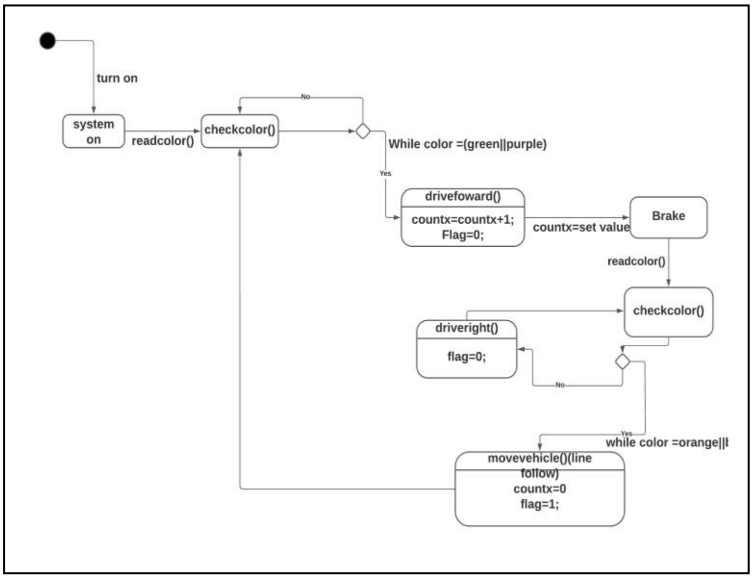


Fig : task 4

5) Drive on the line for two squares, turn 90 degrees and keep driving on the new line for two squares.

For task 5 we made use of some previously used algorithm like the task 3 and task 4 . essentially our vehicle will readcolor and check color and after this depending on the color it will know if its in countx or county, countx being the green and purple line while county being orange and blue. After reading the colors countx or coutn y will begin at zero and count 2 squares by driving forward and using the line follow logic , after this is done, it will brake and start to slowly turning until it reads a different set of colors ,when it does, it will also drive forward and count 2 squares on y , after this the task is completed.

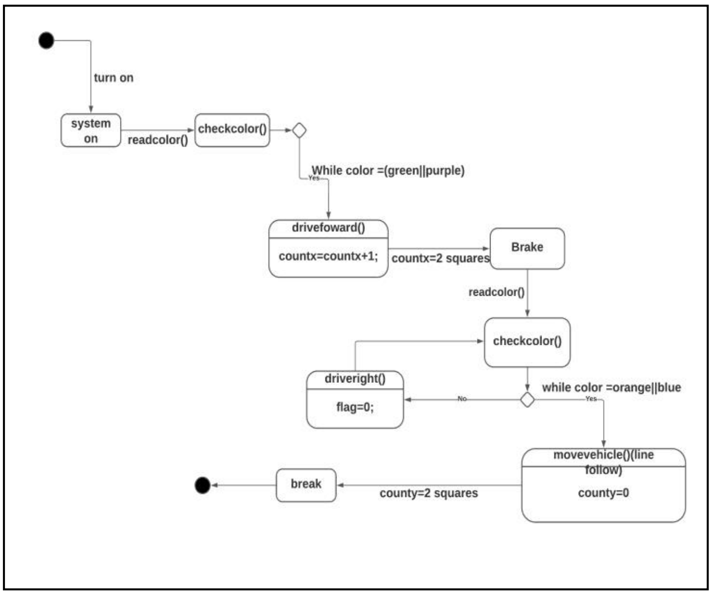
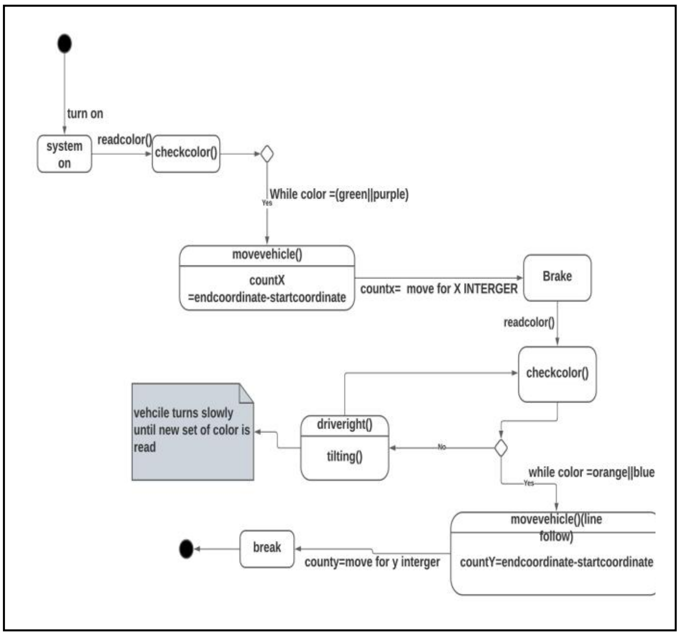


Fig: state machine 5

6) Drive from one point to another (keep track of where you are on the map).

For this part we made an array called end coordinate and start coordinate ,both having x and y values ,the general logic is all about subtracting the end point from the start point to give us an x and y integer to use in our count x and count y holders, as shown in task 5 this will be the main way we move to certain location on the tracks as we will provide the amount of squares needed to be read on both x and y axis .The color sensor reads the amount of steps in both x and y axis while braking and turning where it needs too. For instance, moving from [0,0] to [2,2] will be 2 squares in countx and 2 steps in county. This is how we modeled our system. It is also important to note that the line follow will always help us stay on the track.


turn on
system on
readcolor()
checkcolor()
While color =(green||purple)
Yes
movevehicle()
countX =endcoordinate-startcoordinate
countx= move for X INTERGER
Brake
readcolor()
checkcolor()
vehcile turns slowly until new set of color is read
driveright()
tilting()
No
while color =orange||blue
Yes
movevehicle()(line follow)
countY=endcoordinate-startcoordinate
county=move for y interger
break

7) Drive from point to point, you will get the points to drive to (including the starting point) in a multidimensional array.

Our logic for this part is quite similar to the task 6 we will create a holder for the 3rd or 4th intended coordinate and simply continue to subtract the start point from the end point this will continue as long as we have not reached our desired location,we will be storing the first resultant coordinate as intemediarycoordinate, for example, endcordinateA and endcordinateB will be 2 end points , the first x and y values will be gotten from subtracting startcordinate from endcoridnateA this will give us intermediarycoordinate our final x and y values will be gotten from subtracting the intermediarycooridnates from the start coordinates.

ACKNOWLEDGMENT

We would like to appreciate our prototyping lecturers Kristian Rothers and Gido Wahrmann for their guidance in direction throughout or lab session and design, we would also like to appreciate Professor Dr Henkler for his initial Lectures on system engineering gave us the required knowledge to complete our task.

CONTRIBUTION

THE SYSTEMS ENGINEERING PART OF THIS DOCUMENTATION WAS DONE BY ASHRAF SIDDIQI MOHAMMAD.

THE DESIGN PART OF THIS DOCUMENTATION WAS DONE BY KUYE DOLUWAMU TAIWO

THE SOFTWARE PART WAS CARRIED OUT BY YASHODHAN VISHVESH DESHPANDE

We all edited the document together.

SUBIR BALO MADE ZERO CONTRIBUTIONS TO OUR GROUP AND WAS ABSENT FROM THE FIRST DAY OF CLASS HE MADE NO ATTEMPT TO COOMUNICATE DESPITE BEING PRESENT ON OUR GROUP CHAT.

GITHUB LINK

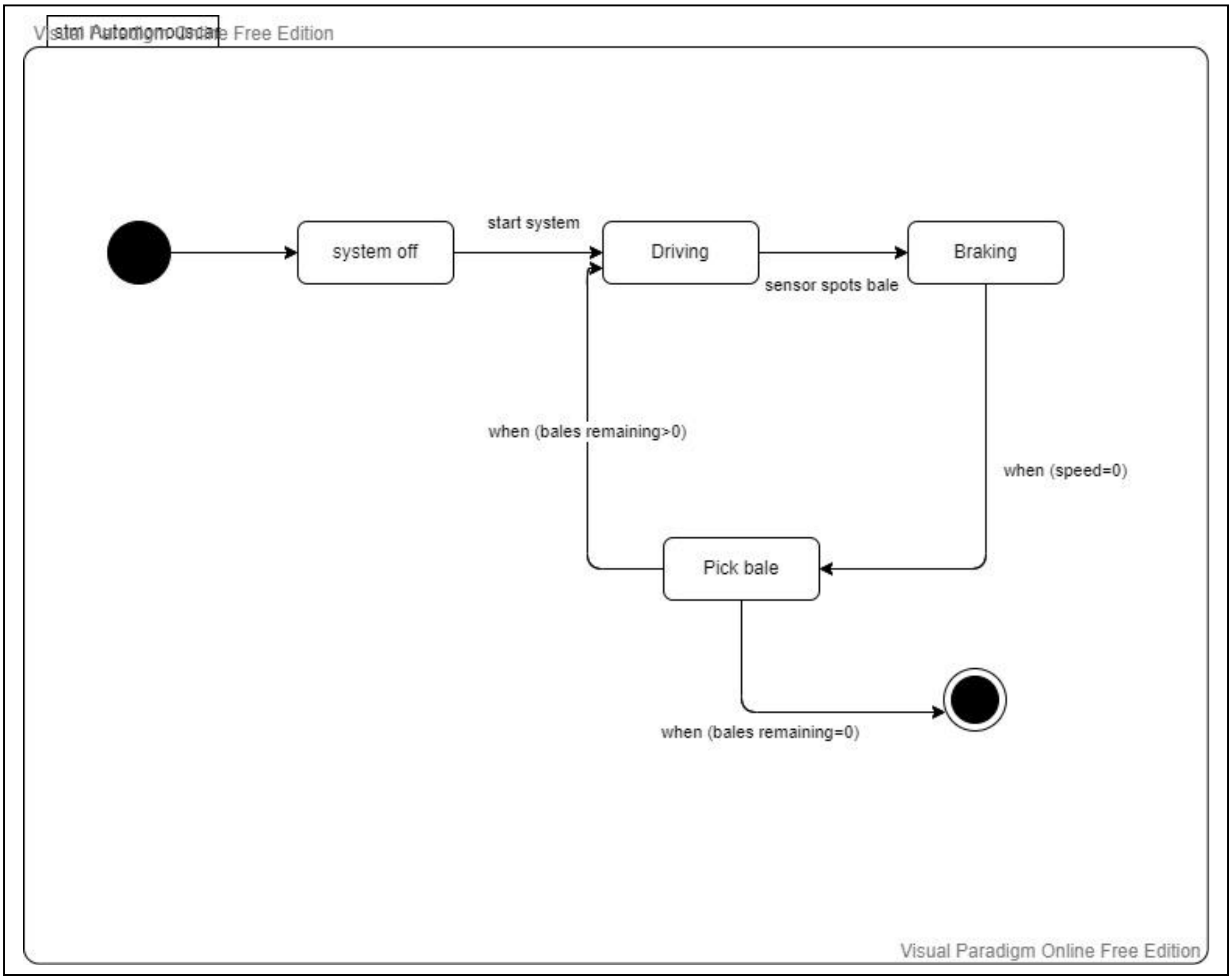
Release=https://github.com/Yashodhandesh/B6_Prototyping_2022/releases

Main github=

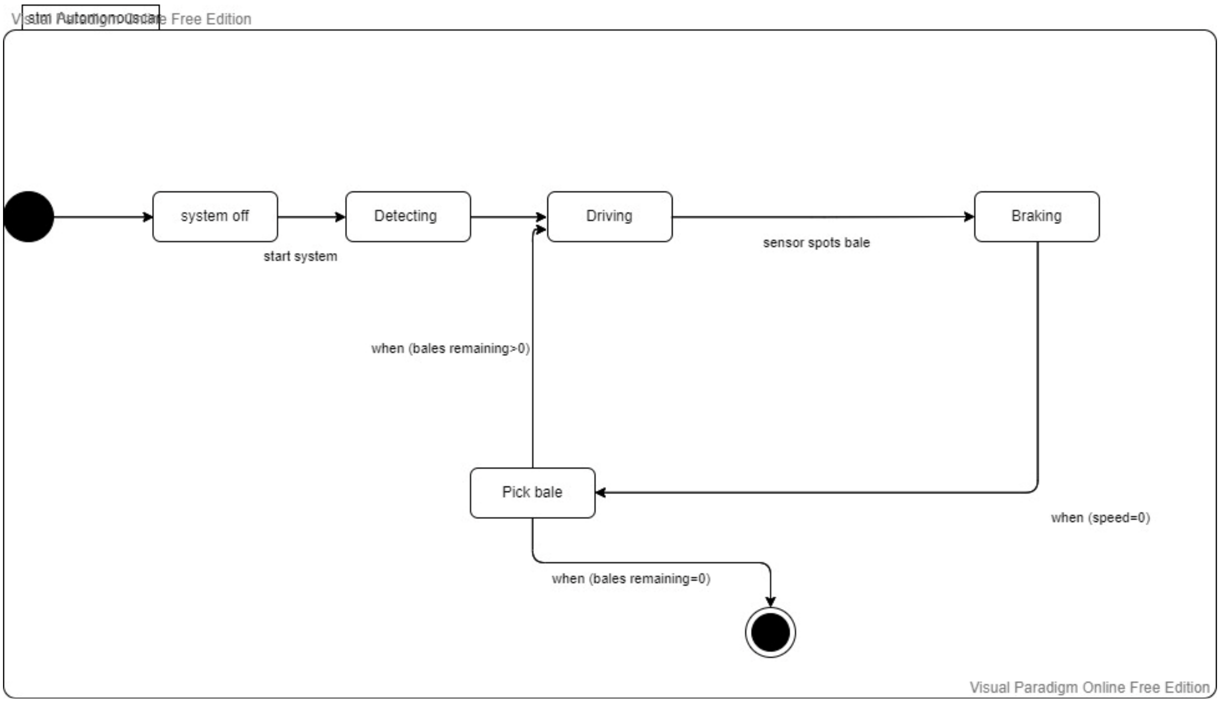
https://github.com/Yashodhandesh/B6_Prototyping_2022.git

APPENDIX

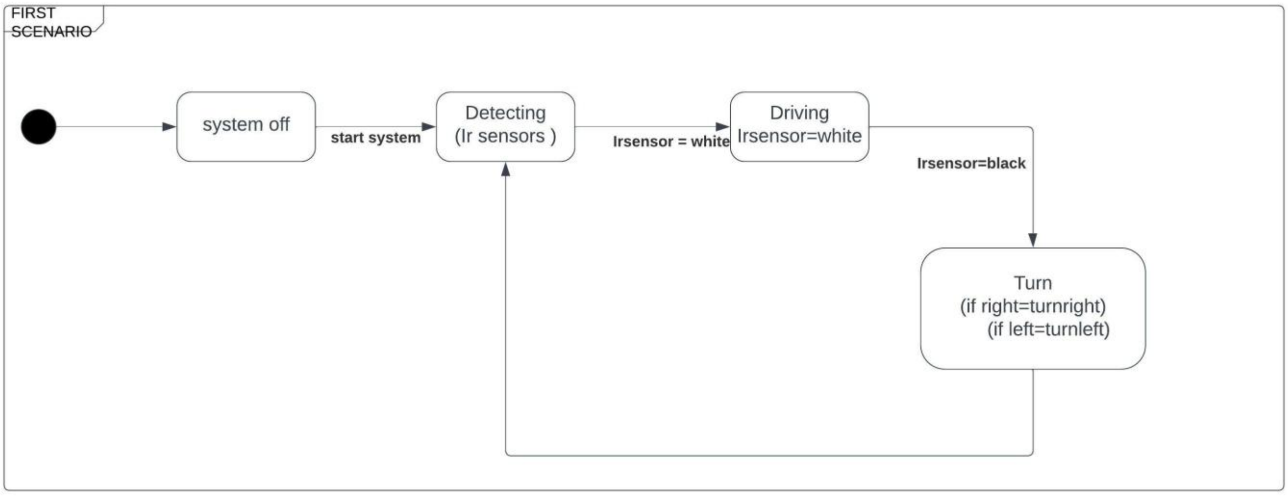
a) FIRST STATE MACHINE



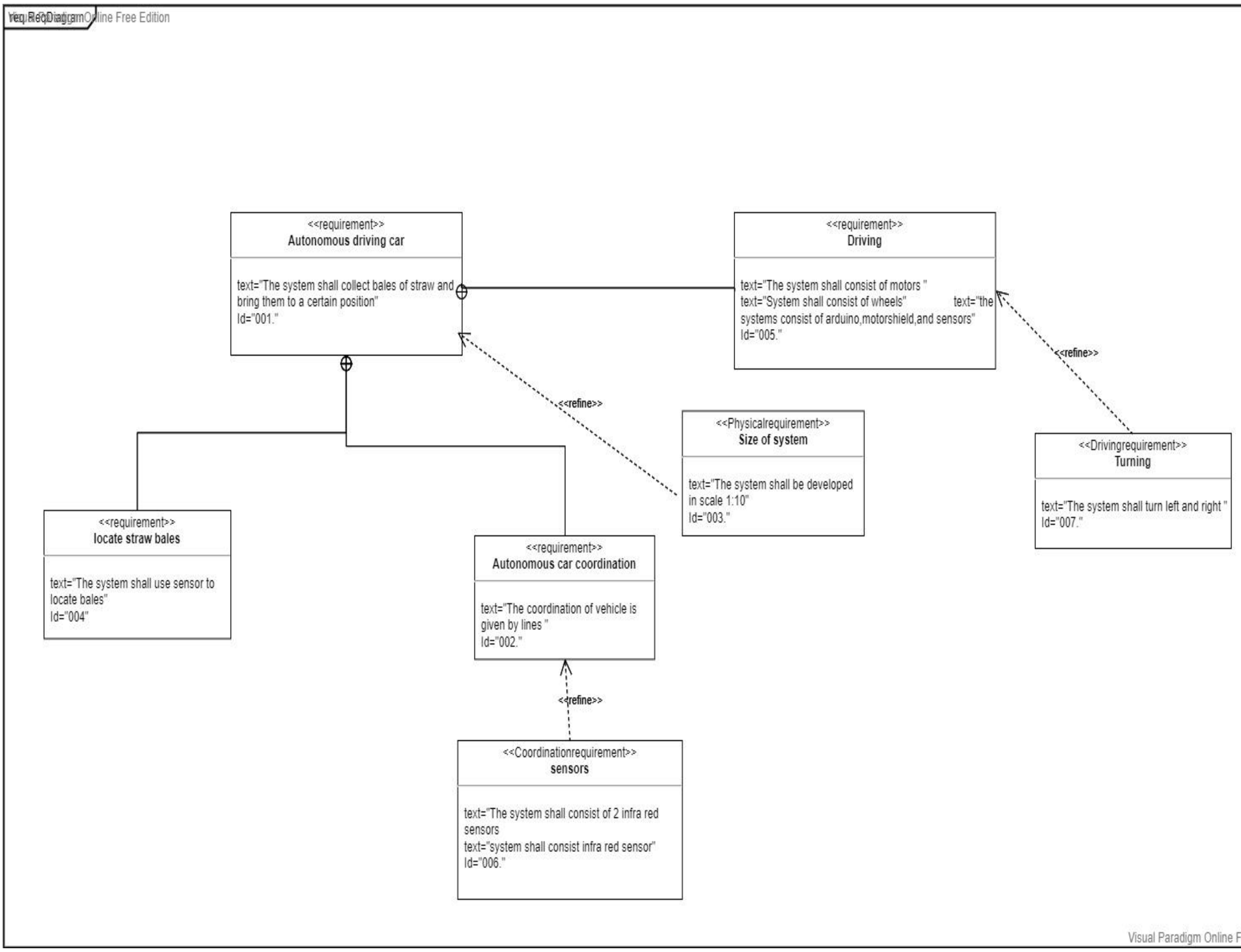
b) 2^ND^ STATE MACHINE DIAGRAM



c) 3^RD^ STATE MACHINE



d) 2ND REQUIREMENT DIAGRAM


req ReqDiagram
Visual Paradigm Online Free Edition
<<requirement>>
Autonomous driving car
text="The system shall collect bales of straw and bring them to a certain position"
Id="001."
<<requirement>>
Driving
text="The system shall consist of motors "
text="System shall consist of wheels" text="the systems consist of arduino,motorshield,and sensors"
Id="005."
<<refine>>
<<refine>>
<<Physicalrequirement>>
Size of system
text="The system shall be developed in scale 1:10"
Id="003."
<<Drivingrequirement>>
Turning
text="The system shall turn left and right "
Id="007."
<<requirement>>
locate straw bales
text="The system shall use sensor to locate bales"
Id="004"
<<requirement>>
Autonomous car coordination
text="The coordination of vehicle is given by lines "
Id="002."
<<refine>>
<<Coordinationrequirement>>
sensors
text="The system shall consist of 2 infra red sensors
text="system shall consist infra red sensor"
Id="006."
Visual Paradigm Online F