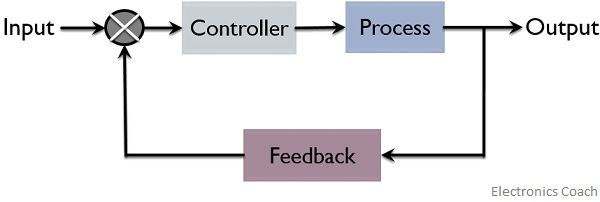
**CONTROL SYSTEM:**

Our system is a simple closed loop system with an input and a microcontroller and an output



**How does the system work?**

**2 IR s**ensors (left &right) are used to determine direction of the black line and give feedback to the microcontroller who in return gives proper commands to the actuators to move in the required directions

The microcontroller is the brain of the system and it has the responsibility to control everything in the system according to the code burn onto it.

To be able to make control on anything we have to understand how it works first to interface it with the microcontroller.

The microcontroller used in the project was AVR Atmega 32 and we interfaced it with the other components in our system

First thing did was to initialize the microcontroller to work in the system and I mean that identifying what are the inputs and the outputs of the system. For this purpose, I used a function called Pin\_direction()

The implementation of the function is written according to the DATASHEET

Of the Atmega32 .

this the implementation of Pin\_direction():

void pin\_direction( char DDrX, char pin, char state)

{

if (state==output)

{

pin\_set((DDrX), pin);

}

else

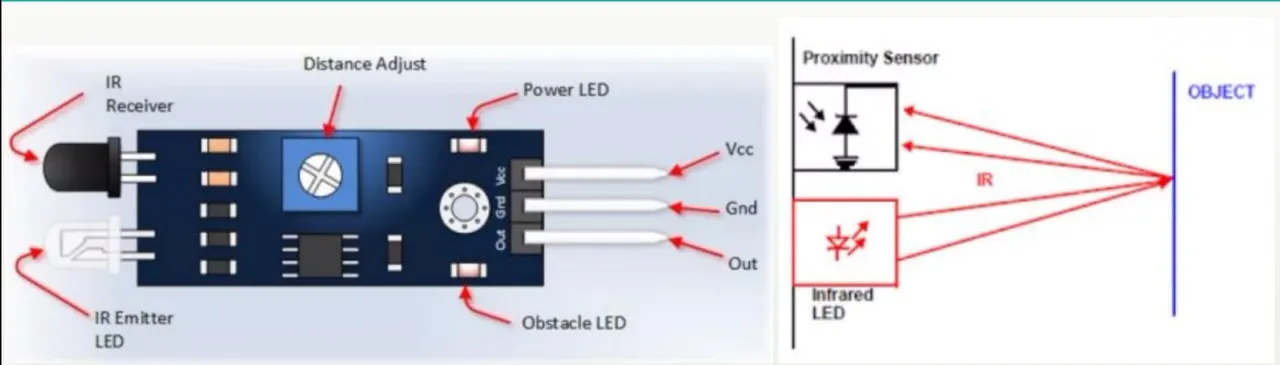
pin\_reset((DDrX), pin);

}

To understand more about how to configure the avr and how to write the function you can read the AVR ATMEGA32 DATASHEET.

The pins connected to the sensors must be input to determine the reading of the sensor but first how does the sensor work and how can we use the working principle to determine what we want (if there is a black line)?

IR sensor or Active infrared sensors both emit and detect infrared radiation. ... When an object comes close to the sensor, **the infrared light from the LED reflects off of the object and is detected by the receiver**. Active IR sensors act as proximity sensors, and they are commonly used in obstacle detection systems



If the receiver detects the signal, it means that there is an object in front of the sensor and it sets the pin (Out) to high which is connected to the microcontroller

There is one special case:

If the object detected was **black** it absorbs the signal an does not reflect it back so the reciever does not receive any signal and so the pin (Out) is not set to high

From this principle we only have to check the pin of the microcontroller (which is connected to the sensor) to determine the direction of the black line

To do this I used the function called Pin\_Read()

The implementation of the function is as follow :

char pin\_read(char pinX,char pin)

{

char x;

x= (\*(volatile char\*)(pinX))&(1<<pin);

return x;

}

This function was also written according to the DATASHEET you can read it to understand more.

The value returned by this function is either 0 or !0 if it is zero it means there is a black line an so we did to determine the direction.

As we mentioned we have 2 sensors so we have 4 cases

Right sensor Left sensor

0 0 //both sensors detect black line

0 1 //right sensors detect black line

1 0 //left sensors detect black line

1 1 // no black line

In each case of the previous cases the microcontroller must make a different decision according to the code and here we have to interface the AVR with DC motor

For this purpose, as mentioned we used a motor driver and the reasons are discussed above …

To make the motor rotate and we only have to set the pin of the driver which connected to the microcontroller and I used the function Pin\_set() to do this

The implementation of this function is as follow:

void pin\_set(char reg,char pin)

{

\*((volatile char\*)(reg))|=(1<<pin);

}

Go to DATASHEET to understand more …

1st Case : both detect black line :

The microcontroller will give command to the driver to stop the motors

2nd Case : right sensor detect black line :

The microcontroller will give command to the driver to turn right

And this done by stopping left motor and right one is rotating

3rd Case : left sensor detect black line :

The microcontroller will give command to the driver to turn left

And this done by stopping left motor and left one is rotating

4th Case : on black line :

The microcontroller will give command to the driver tomove forward.

**CODE:**

The code was written in C programming language and is divided into 2 main parts

1.First part:

Initialization of the robot and this done only one time when executing the code.

I mean by initialization that I identify what are the inputs and outputs of the system

Sensors are the inputs because the detect the line and give feedback to microcontroller, motors are the output of the system, they receive commands from the microcontroller to move in the proper direction.

To initialize I wrote a function Initialize\_The\_Robot();

This the implementation of the function:

Initialize\_The\_Robot()

{

pin\_direction(Motors\_DDr,Right\_Motor,output);

pin\_direction(Motors\_DDr,Left\_Motor,output);

pin\_direction(Sensor\_DDr,Right\_sensor,input);

pin\_direction(Sensor\_DDr,Left\_sensor,input);

}

2.Second part:

Follow the line part , here we have to handle the 4 conditions of the sensors discussed . This done as long there is power in the circuit.

I made a function called Follow\_The\_Line();

This the implementation of this function:

Follow\_The\_Line()

{

char left=pin\_read(Sensor\_Pin,Left\_sensor);

char right=pin\_read(Sensor\_Pin,Right\_sensor);

if((right!=0)&&(left!=0))

{

Move\_forward();

}

else if((right==0)&&(left!=0))

{

Turn\_left();

}

else if((right!=0)&&(left==0))

{

Turn\_right();

}

else

{

Stop();

}

}

After handling the different condition and making the implementation of the fuctions used in our programm we come to the MAIN PROGRAMM

#include "GPIO\_AVR.h"

#include "GPIO\_AVR\_PORTS.h"

#include "Line follower.h"

int main(void)

{

Initialize\_The\_Robot();

while (1)

{

Follow\_The\_Line();

}

}

To make my code that easy I divided into many files either header files (.H) or C file (. C)

1.first file(line follower.h):

#define Right\_Motor 1

#define Left\_Motor 6

#define Right\_sensor 0

#define Left\_sensor 7

#define Motors\_Port portD

#define Sensor\_Pin pinC

#define Motors\_DDr DDrD

#define Sensor\_DDr DDrC

2.second file (line follower .c)

Move\_forward()

{

pin\_set(Motors\_Port,Left\_Motor);

pin\_set(Motors\_Port,Right\_Motor);

}

Turn\_right()

{

pin\_set(Motors\_Port,Right\_Motor);

pin\_reset(Motors\_Port,Left\_Motor);

}

Turn\_left()

{

pin\_reset(Motors\_Port,Right\_Motor);

pin\_set(Motors\_Port,Left\_Motor);

}

Stop()

{

pin\_reset(Motors\_Port,Right\_Motor);

pin\_reset(Motors\_Port,Left\_Motor);

}

3.Third file (GPIO\_AVR\_PORTS.h): in this file I accessed each rigister according to its address from AVR DATASHEET

#define output 1

#define input 0

#define pinA 0x39

#define portA 0x3B

#define DDrA 0x3A

#define pinB 0x36

#define portB 0x38

#define DDrB 0x37

#define pinC 0x33

#define portC 0x35

#define DDrC 0x34

#define pinD 0x30

#define portD 0x32

#define DDrD 0x31