A QUICK ON OBSTACLE AVOIDING ROBOT

INDEX

1 Abstract:

This is a quick guide for making and understanding line follower robot including PWM and PID control which will avoid obstacle while following the line. When the obstacle is detected on its path it will decide which path to proceed.

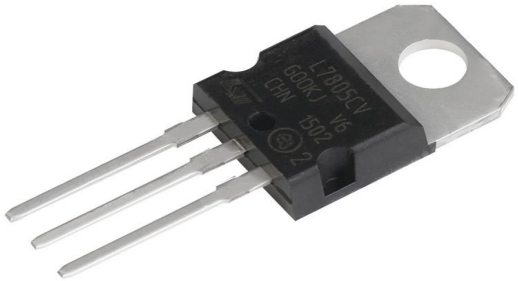
2 List of components used:

- GCB
- Male-female headers
- LED
- Resistor (1 kiloohm)
- Voltage regulator (LM7805)
- Capacitors (1&10 microfarad)
- Array line sensor
- Screw connector
- Switch
- Atmega32 MCU
- Atmega32 IC bed
- USBASP Programmer
- FRC port
- 12V battery
- Jumper wires
- L293D +IC bed
- Castor wheel
- Wheels
- Multi-strand wires
- 2 motors (300 rpm)
- Heat sink
- Plywood (Readymade Chassis)
- Solder wire
- Screws
- Screw holders
- Hc sr04 (ultrasonic sensor)

3 Hardware implementation

3.1 IC LM7805 (Voltage Regulator)

Voltage sources in a circuit may result in fluctuations of voltage outputs. This IC maintains the output voltage at a constant value. 7805 IC, a member of 78xx series of fixed linear voltage regulators used to maintain such fluctuations . The xx in 78xx indicates the output voltage it provides. 7805 IC provides +5 volts regulated power supply with provisions to add a heat sink.



RATING:

- Input voltage range 7V- 35V
- Current rating $I_{c=}$ 1A
- Output voltage range $V_{Max=5.2V}$, $V_{Min=4.8V}$

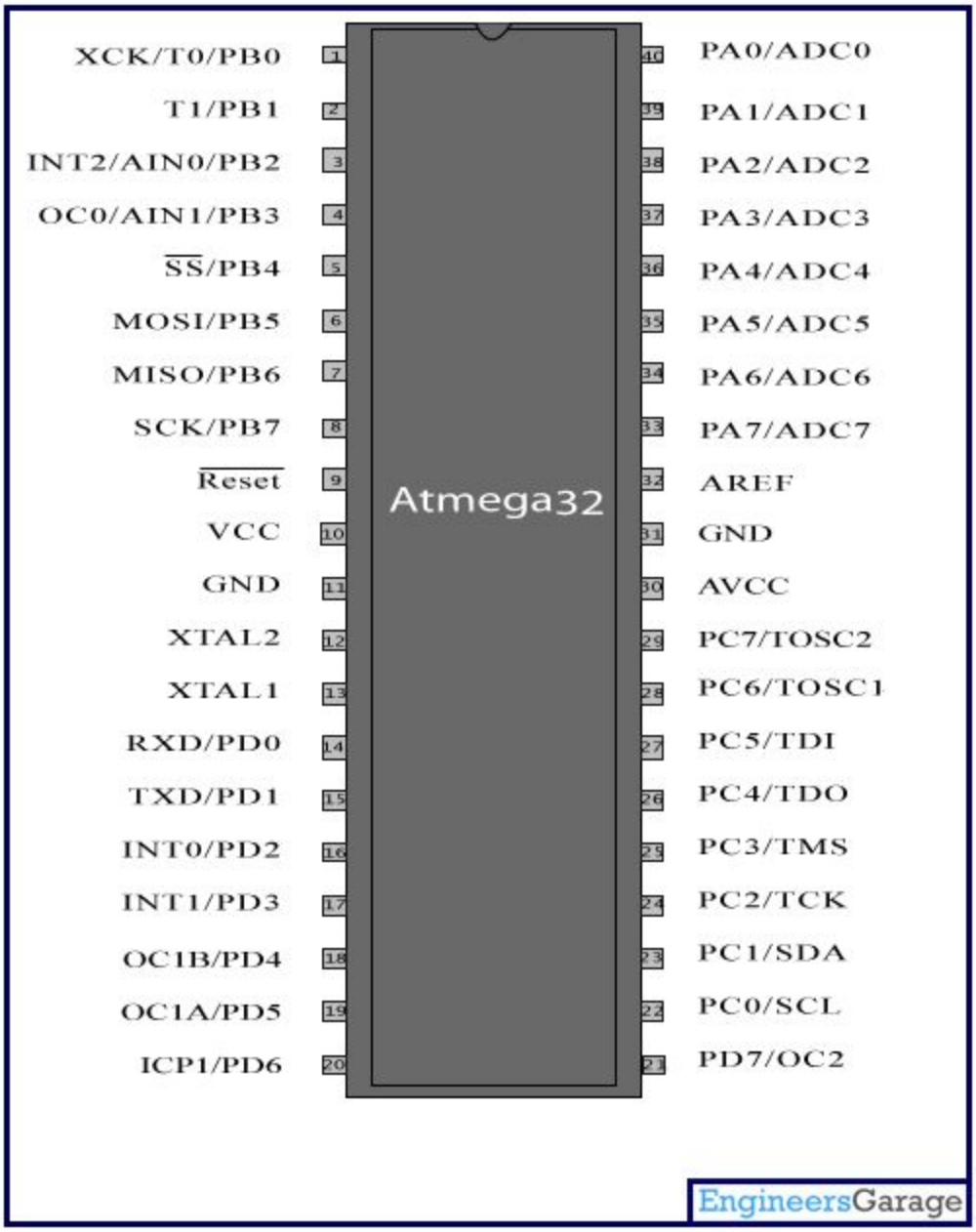
Pin Details:

	Pin	Function	Description
	INPUT	Input voltage (7V-35V)	In this pin of the IC positive unregulated voltage is given in regulation.
	GROUND	Ground (0V)	In this pin where the ground is given. This pin is neutral for equally the input and output.
	OUTPUT	Regulated output; 5V (4.8V-5.2V)	The output of the regulated 5V volt is

			taken out at this pin of the IC regulator.

3.2 ATMEGA32 AVR MICROCONTROLLER

PIN DIAGRAM:



Port A (PA7..PA0)

Port A serves as the analog inputs to the A/D Converter.Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.

Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit).

Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit).

Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source

current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

FEATURES:

- Operating Voltage: **4.5V to 5V**
- Advanced RISC Architecture.
- 32 x 8 General Purpose Working Registers.
- Fully Static Operation.
- Up to 16 MIPS Throughput at 16 MHz.
- On-chip 2-cycle Multiplier.
- High Endurance Non-volatile Memory segments.
- 32 Kbytes of In-System Self-programmable Flash program memory.
- 1024 Bytes EEPROM.
- 2 Kbyte Internal SRAM.
- Write/Erase Cycles: **10,000 Flash/100,000 EEPROM**.
- Data retention: 20 years at **85°C/100** years at **25°C(1)**.
- Optional Boot Code Section with Independent Lock Bits.
- In-System Programming by On-chip Boot Program.
- True Read-While-Write Operation.
- Programming Lock for Software Security.
- Extensive On-chip Debug Support.
- 8 Single-ended Channels.
- Byte-oriented Two-wire Serial Interface.
- Programmable Serial USART.
- Master/Slave SPI Serial Interface.
- Programmable Watchdog Timer with Separate On-chip Oscillator.

3.3 Motor Driver IC(L293D)

· A motor driver IC is an integrated circuit chip which is usually used to control motors in autonomous robots. Motor driver ICs act as an interface between microprocessors in robots and the motors in the robot.

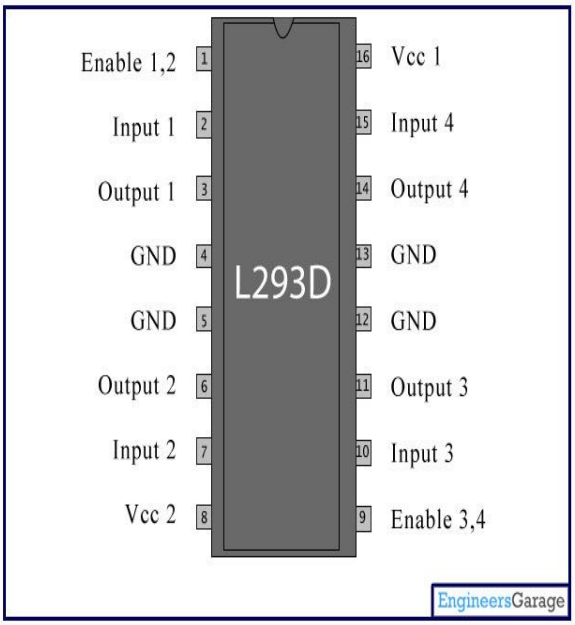
· The most commonly used motor driver IC's are from the L293 series such as L293D, L293NE, etc. These ICs are designed to control 2 DC motors simultaneously. Some motor driver like L298 is designed to control only one DC motor .

Why we need Motor Driver IC?

· Motor Driver ICs are primarily used in autonomous robotics only. Also most microprocessors operate at low voltages and require a small amount of current to operate while the motors require a relatively higher voltages and current .

· Thus current cannot be supplied to the motors from the microprocessor. This is the primary need for the motor driver IC.

Pin diagram of L239D:-



Pin No. Pin Characteristics

- **1** Enable 1-2, when this is HIGH the left part of the IC will work and when it is low the left part won't work. So, this is the Master Control pin for the left part of IC
- **2** INPUT 1, when this pin is HIGH the current will flow though output 1
- **3** OUTPUT 1, this pin should be connected to one of the terminal of motor
- **4,5** GND, ground pins
- **6** OUTPUT 2, this pin should be connected to one of the terminal of motor
- **7** INPUT 2, when this pin is HIGH the current will flow though output 2
- **8** VC, this is the voltage which will be supplied to the motor. So, if you are driving 12V DC motors then make sure that this pin is supplied with 12 V
- **16** VSS, this is the power source to the IC. So, this pin should be supplied with 5 V
- **15** INPUT 4, when this pin is HIGH the current will flow though output 4
- **14** OUTPUT 4, this pin should be connected to one of the terminal of motor
- **13,12** GND, ground pins
- **11** OUTPUT 3, this pin should be connected to one of the terminal of motor
- **10** INPUT 3, when this pin is HIGH the current will flow though output 3
- **9** Enable 3-4, when this is HIGH the right part of the IC will work and when it is low the right part won't work. So, this is the Master Control pin for the right part of IC

How Motor Driver Operates?

- The L293D IC receives signals from the microcontroller and transmits the relative signal to the motors. It has two voltage pins, one of which is used to draw current for the working of the L293D and the other is used to apply voltage to the motors. The L293D switches it output signal according to the input received from the microcontroller.
- For Example: If the microcontroller sends a 1(digital high) to the Input Pin of L293D, then the L293D transmits a 1(digital high) to the motor from its Output Pin.
- An important thing to note is that the L293D simply transmits the signal it receives. It does not change the signal in any case.

3.4 MOTOR(VEGA 300 rpm)

Features:

- 300RPM 12V DC motors with Gearbox
- 4mm shaft diameter with internal hole
- 125gm weight
- Same size motor available in various rpm
- 2kg-cm torque
- No-load current = 60 mA(Max), Load current = 300 mA(Max)

3.5 BATTERY

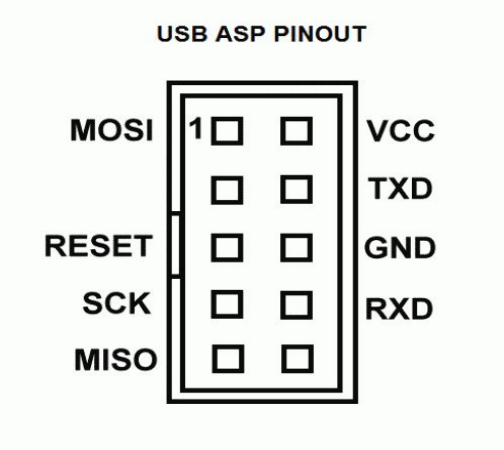
Nominal Capacity:
2200mAh
Nominal Voltage:
11.1V
Type:
Li-Polymer Battery
Application:
RC Car Boat Truck Heli Airplane
Weight:
186g
Cycle life:
500 Times
Discharge Rate:
25C
Max Burst Discharge Rate (C):
50C

Connector Type:
XT60 Deans T Plug

3.6 FRC PORT

The orientation of the FRC cable on both sides should be kept in mind with the help of the
nodge given on its black part.
Visit this link for a detailed explanation of setting up Atmel Studio, installing USBASP

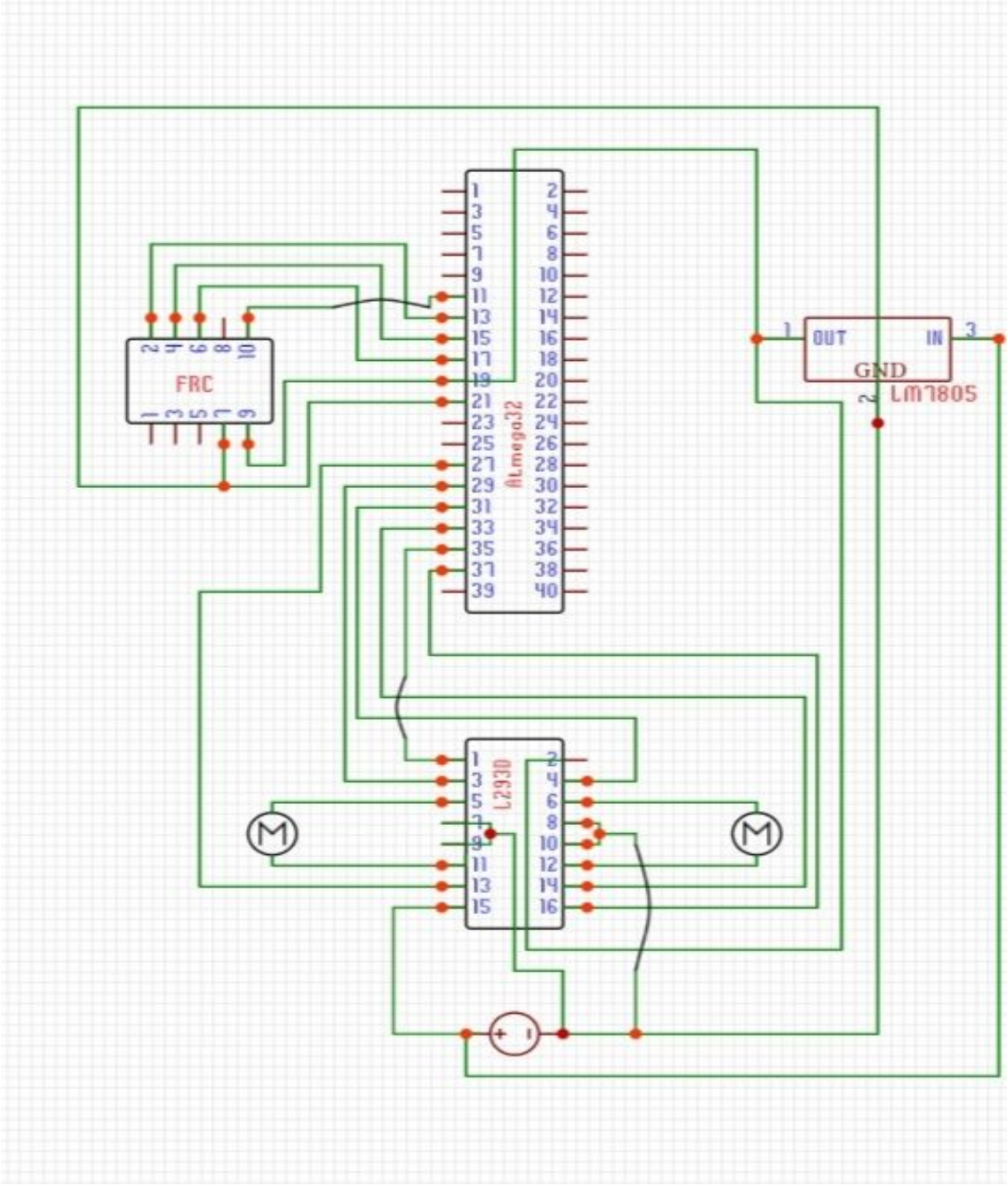
PINOUT:



Why do you need USBASP Programmer?

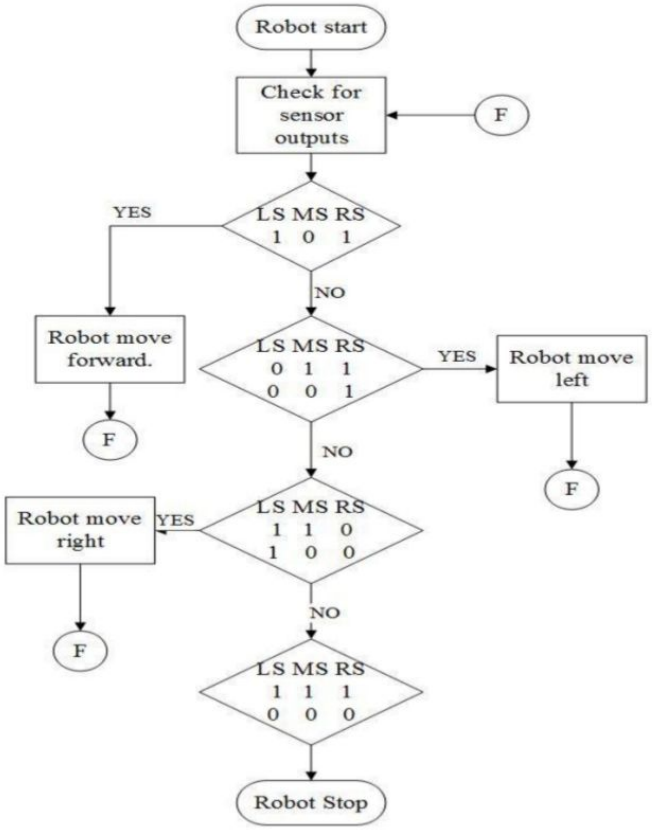
Simple answer to this question is to burn the program written in our software to our microprocessor.It can be used for a wide range of AVR microprocessor. It simply consists of an ATMega88 or an ATMega8 and a couple of passive components. The programmer uses a firmware-only
USB driver, no special USB controller is needed.

4 Circuit diagram



5 FLOW CHART OF LINE FOLLOWER

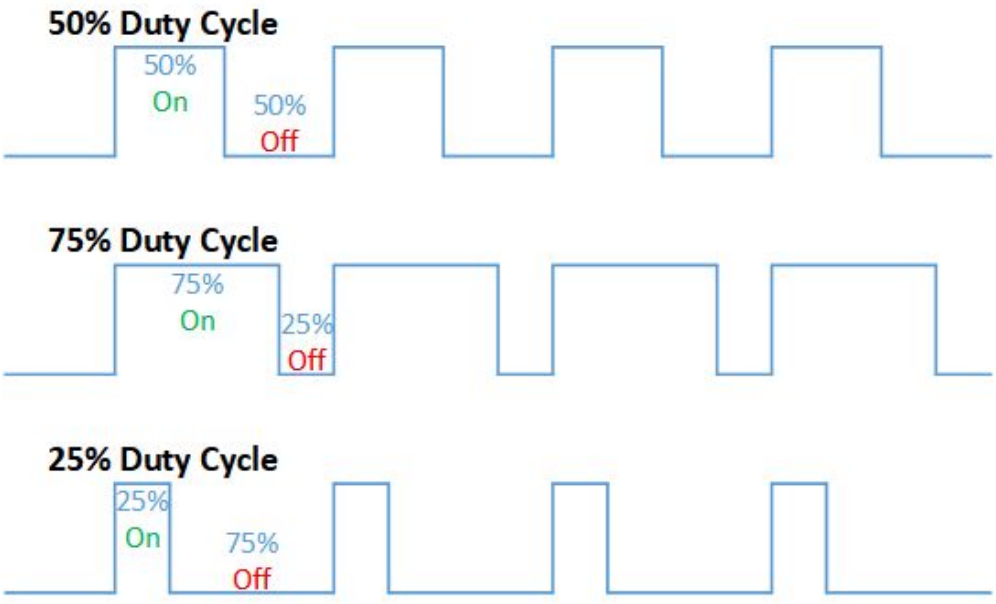
This flow-chart explains how the robot movement related to the sensor inputs.



6 Pulse Width Modulation

PWM is used to gain control over output voltages. This allows us to drive the motor at different rpms by varying the supply voltage to it thus allowing better control over the robot especially while turning. PWM is implemented by rapidly switching ‘On’ and ‘Off’ the output voltage which has an averaging effect thus producing an intermediate value of voltage.

The term duty cycle describes the proportion of 'on' time to the regular interval or 'period' of time; a low duty cycle corresponds to low power, because the power is off for most of the time. Duty cycle is expressed in percent, 100% being fully on. When a digital signal is on half of the time and off the other half of the time, the digital signal has a duty cycle of 50% and resembles a "square" wave. When a digital signal spends more time in the on state than the off state, it has a duty cycle of >50%. When a digital signal spends more time in the off state than the on state, it has a duty cycle of <50%. Here is a pictorial that illustrates these three scenarios:



Implementing PWM:

1. A Waveform is Generated by modifying the WGM bits on the TCCR0 Register

```
TCCR0|=(1<<WGM00)|(1<<WGM01);//Sets the PWM
```

2. The PWM wave is then generated in an inverting mode

```
TCCR0|=(1<<COM00)|(1<<COM01); //Set inverting mode
```

3. The ICP1 pin will be set as a timer counter. The reference will be set at 1000 counts for better pwm control. This requires us to prescale the 255 bit counter so it can count upto 1000.

```
TCCR0|=(1<<CS01); //Prescale 8 bit counter to prescaling of 8
ICR1=1000; //Sets ICP1(Reference to 1000)
```

4. The pwm output will given to OC1 and OC2 Pins of the microcontroller. These are set with reference to the ICP1. For Example:

```
OCR1A=450;//Outputs 1 at 45% of max value
OCR1B=1000;//Outputs at 100% of max value
```

7 PROPORTIONAL INTEGRAL DERIVATIVE(PID)

A proportional–integral–derivative controller (PID controller or three-term controller) is a control loop feedback mechanism widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an *error value* {e(t)} as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted *P*, *I*, and *D* respectively), hence the name.

In practical terms it automatically applies accurate and responsive correction to a control function. An everyday example is the cruise control on a car, where external influences such as hills (gradients) would decrease speed. The PID algorithm restores from current speed to the desired speed in an optimal way, without delay or overshoot, by controlling the power output of the vehicle's engine.

In this model:

- Term P is proportional to the current value of the SP − PV error e(t). For example, if the error is large and positive, the control output will be proportionately large and positive, taking into account the gain factor "K". Using proportional control alone in a process with compensation such as temperature control, will result in an error between the setpoint and the actual process value, because it requires an error to generate the proportional response. If there is no error, there is no corrective response.

- Term I accounts for past values of the SP − PV error and integrates them over time to produce the I term. For example, if there is a residual SP − PV error after the application of proportional control, the integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error. When the error is eliminated, the integral term will cease to grow. This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect.

- Term D is a best estimate of the future trend of the SP − PV error, based on its current rate of change. It is sometimes called "anticipatory control", as it is effectively seeking to reduce the effect of the SP − PV error by exerting a control influence generated by the rate of error change. The more rapid the change, the

greater the controlling or dampening effect.

In our project,our desired setpoint is that our bot should exactly be lying upon the white line i.e the center of robot should coincide with the center of the line(the center leds should always be high).
And thus the error will be created when the bot is not exactly on the line.
Now ,for the calculation of error,certain values should be assigned to the respective LED's.
Initially we have given the following values.

```
int pin_weightage[8]={-4,-3,-2,-1,1,2,3,4};//keeping that in mind that when the bot is in center the error will be zero
```

But the problem was that when all the pins were high then also the error will be zero.

Therefore this weightage was used..

```
int pin_weightage[8]={0,2,4,6,8,10,12,14};
```

```
for(i=0;i<8;i++)
{
    if(bit_is_set(PINA,i)==1)
    c++;//a counter to count the total high pins.

error+=(bit_is_set(PINA,i)*pin_weightage[i]);//this was done to calculate the net error caused by the number of high pins.
    }
```

error=error/c;
This was done to make the net error in 0 to maximum form i.e in continuously increasing manner.

NOTE: We have to insert that condition when all pins are low then make c=1,to prevent runtime error during execution.

error=error-desired_value;
To calculate the net error and in this case the desired value should be 7 as when the bot will be in center then the error as per calculations would be 7 , therefore making net 0.

IMPLEMENTATION OF P CONTROL

For this purpose a constant is defined as kp whose value totally depends upon the hardware of system and as the name suggests the net error according to p control will be kp*error.

Value of kp will be figured out by hit and trial method.

IMPLEMENTATION OF D CONTROL

For this purpose a constant is defined as kd whose value totally depends upon the hardware of system and as the name suggests the net error will be dependent upon the difference of the present error and the last error calculated.
Value of kd will be figured out by hit and trial method.

error1+=kp*error+kd*d_error;//the total net error

d_error=error1-last_error;//the difference of two errors.

last_error=error1;//storing the last error

Now what if the bit is off track,
Therefore , any optimum maximum value will be
assigned to the error when it is off track.

Proportional Mode Advantages

- Minimize dead time from stiction and backlash
- Minimize rise time
- Minimize peak error
- Minimize integrated error

Proportional Mode Disadvantages

- Abrupt changes in output upset operators
- Abrupt changes in output upset other loops
- Amplification of noise

Integral Mode Advantages

- Eliminate offset
- Minimize integrated error
- Smooth movement of output

Integral Mode Disadvantages

- Limit cycles
- Overshoot
- Runaway of open loop unstable reactors

Derivative Mode Advantages

- Minimize dead time from stiction and backlash
- Minimize rise time
- Minimize peak error
- Minimize integrated error

Derivative Mode Disadvantages

- Abrupt changes in output upset operators
- Abrupt changes in output upset other loops
- Amplification of noise

8 ULTRASONIC SENSOR

Ultrasonic Ranging Module HC - SR04 Product features: Ultrasonic ranging module HC - SR04 provides 2cm - 400cm ideally non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
 (2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
 Test distance = (high level time×velocity of sound (340M/S) / 2, Wire connecting direct as following: 5V Supply Trigger Pulse Input Echo Pulse Output 0V Ground Electric Parameter Working Voltage DC 5 V Working Current 15mA Working Frequency 40Hz Max Range 4m Min Range 2cm Measuring Angle 15 degree Trigger Input Signal 10uS TTL pulse Echo Output Signal Input TTL lever signal and the range in proportion Dimension 45*20*15mm

EXECUTION CODE:

1. Enable interrupt to 0

```
GICR|=(1<<INT0);
```

2. Set interrupt to trigger logic change

```
MCUCR|=(1<<ISC00);
```

3. Store the digital output

```
int16_t COUNTA = 0;
```

4. Enable the global interrupts

```
sei();// enabling global interrupts
```

5. It will trigger the interrupt function {ISR(INT0_vect) }

```
PORTD|=(1<<PIND0);
```

6.It will trigger the sensor for 15 microsec

```
_delay_us(15);
```

7.Changing the logic level will again trigger the interrupt function.

```
PORTD &=~(1<<PIND0);
```

8.Interrupt service routine when there is a change in logic level.

```
ISR(INT0_vect)
```

9. Enabling and Disabling counter

```
{
        if (i==1)//when logic from HIGH to LOW
        {
                TCCR1B=0;//disabling counter
                pulse=TCNT1;//count memory is updated to integer
                TCNT1=0;//resetting the counter memory
                i=0;
        }

        if (i==0){//when logic change from LOW to HIGH
                TCCR1B|=(1<<CS10);//enabling counter
                i=1;
        }
}
```

After the count is calculated then the net distance is found by using the formula

COUNTA = (pulse/58);//getting the distance based on formula on introduction

And if the counta value is less than 10 then our bot will take an uturn.

9 Problems faced and their solutions :

P-1 There was connectivity problem between the atmega and the circuit?
SOL - We were using headers instead of their original bed, due to which there were connectivity issues. When we replaced that it started working normally

P-2. Our I.C. was getting very hot.
SOL -This was because we were using sorted atmega.

P-3. We lost our 3 atmega32 even whole circuit was correct.
SOL -It just happened bcoz we have not checked our motor driver and unfortunately it was already not working properly so We replaced the motor driver and then our problem get solved.

P-4 The predefined function of bit_is_set does not return 1 when it detects some highly reflective surface but rather some non zero value .

10 What we have Implemented :

- Simple line follower
- Line follower with **PWM**
- Line follower with **PID**
- simple obstacle avoiding using single **IR SENSOR**

11 Timeline (22jan-12feb)

3feb - line follower with pwm
5feb - line follower with pid

7feb - line follower with pwm & ultrasonic
10feb - line follower with pid & ultrasonic
12feb - final optimisation