CAPSTONE PROJECT REPORT

(Project Term January-May 2011)

(FIRE FIGHTING ROBOT Design)

DECLARATION

I hereby declare that the project work entitled ("Fire Fighting Robot Design") is an authentic record of my own work carried out as requirements of Capstone Project for the award of degree of B-tech M-tech ECE from Lovely Professional University, Phagwara, under the guidance of (Mr. ABHISHEK KUMAR SRIVASTAVA), during January to May, 2011).

(Signature of student)

Name of Students (1) PANKAJ SONI
Registration no:-3060070085

(2) ANIRUDH NANDA
Registration no:-3060070098

(3) NITTIN SHARMA
Registration no:-3060070093

Date: 25-04-2011

This is to certify that the above statement made by the student is correct to the best of my knowledge and belief.

(Name : Mr. ABHISHEK KUMAR SRIVASTAVA)

Faculty Mentor

ACKNOWLEDGEMENT

There are numerous persons we would like to thank for their contribution to our team's success in the Fire Fighting robot. Nothing concrete can be achieved without an optimal combination of inspiration and perspiration. No work can be accomplished without taking the guidance of the experts. It is only the critiques from ingenious intellectuals that help transform a product into a quality product.

It has been good to work on this project and a great experience in itself.

Name of Students (1) PANKAJ SONI
Registration no:-3060070085

(2) ANIRUDH NANDA
Registration no:-3060070098

(3) NITTIN SHARMA
Registration no:-3060070093

Programme :-B-tech M-tech ECE

Section :-G67M1

LOVELY PROFESSIONAL UNIVERSITY

INDEX

1. Profile of the Problem

To design and construct a Fire-Fighting Robot which would extinguish multiple candles or match sticks whilst avoiding any obstacles in the robot's path. This project was a group project and was to be completed before the end date of 25 May 2011.

The main structure of the robot was made of light aluminum plate. The drive line was provided by two low speed motors with connected gearboxes and two RC wheels. We decided on FIRE SENSOR(thermistor) sensors to provide the distance we were from objects. The brains of our robot were provided by a microcontroller .

Each member of the group was allocated a different section of the project to complete. We had allocated times to work on the project in the lab. The sections we split the robot up into were,, Mechanical, Electrical and final the Software. These areas are discussed in greater detail in the report. We believe the factors influencing the success of the robot were good teamwork and starting the project as soon as practically possible.

On completion of the initial programming of the robot we had a testing process to undertake. The initial testing was conducted in stages. Firstly we needed to get the obstacle avoidance working as required then we incorporated the flame detection circuit at a later date. We conducted a fine tuning session where we adjusted our code to ensure the robot's sensors were operating as efficiently as possible.

We encountered numerous problems with the construction of our robot. We had wheels slipping on the shaft under load, the sensor wires moving during operation (thus not giving a correct reading) and stability problems as we only used two wheels to drive the robot. All problems were overcome in the later stages of then project.

Existing System

Introduction

This year the project was to design and build a Fire-Fighting Robot. The robot was required to move around obstacles and detect and extinguish small fires, which were simulated by candles. The robots were required to be built by the end of semester one with a competition against the other groups as a final assessment.
As to building the robot itself we were also required to complete a report for our robot. These were to give greater explanation of the design and construction phases of the robot as well as to visually show diagrams of the robot we constructed.

Project Specifications

Initially there were a number of specifications that needed to be met in order to successfully complete the project. The fire fighting robot we produced was designed, built and operated with these project specifications in mind. These specifications are listed below:

● The Fire-Fighting Robot is a group project - groups are to be chosen by the students with a maximum of five in each group.

● The robot is to be autonomous when in run mode, and no human intervention is to take place throughout the entire assessment time.

● The robot is required to find and extinguish two candles randomly placed on any surface.

● The robot is to implement obstacle avoidance at all times whilst tracking and extinguishing the candles.

● The robot is not to use any destructive methods to extinguish the candle. The candle is not to be touched by the robot at any time during the assessment.

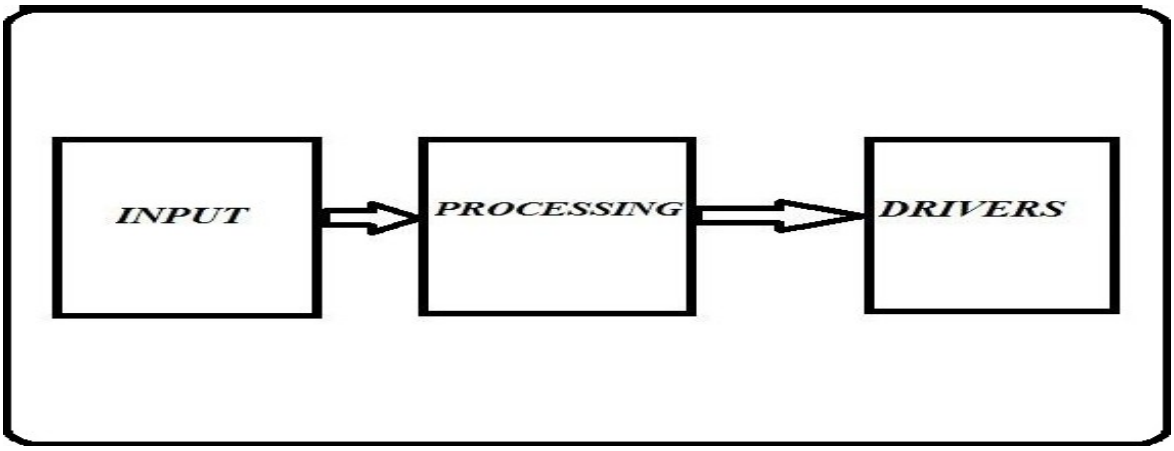
● The size and the dimensions of both the robot and candles is determined by us.

Objective

Many house fires originate when someone is either sleeping or not home so our objective is:-
To avoid the damage caused by fire.
Minimize the air pollution

ROBOTICS:

BASIC STRUCTURE OF ROBOTICS



A mechanical device that sometimes resembles a human and is capable of performing a variety of often complex human tasks on command or by being programmed in advance is defined as a ROBOT

A Robot never becomes fulfill until it can take decisions. We can't go for computers for the computation & intelligence. But we can use single chip microcontrollers for controlling. This robot is not going to do any big tasks there will only be predefined tasks. The robot's full control is embedded into one chip which is otherwise known as embedded systems designing.

To make a robot we must surely know to at least use a single microcontroller. So let us see about Embedded System thru 8051.

What is Open SYSTEM?

An open system is the normal desktop computer where you can use it for any tasks. If you want to process text documents you can install Ms-word if you want to send mail you can use outlook express and counts on.

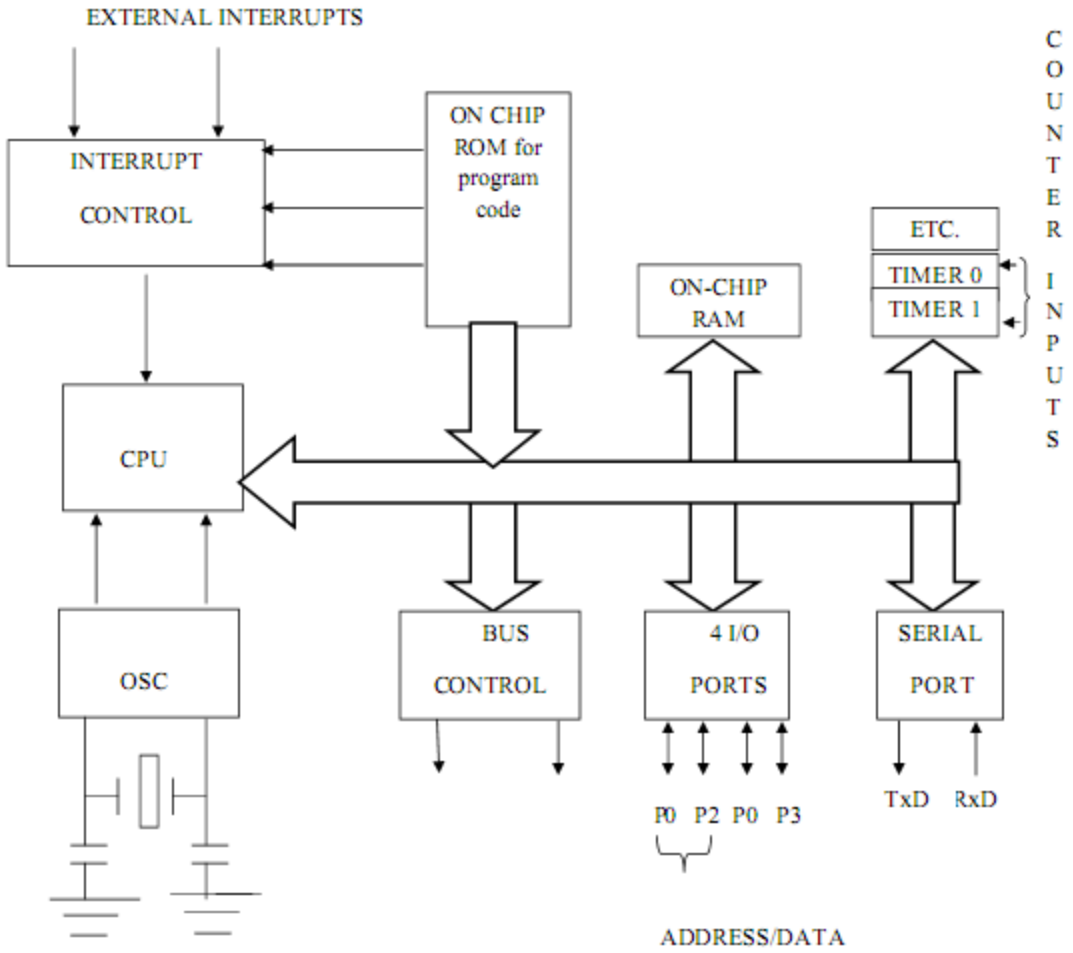
What is Embedded System ?

An embedded system is the system where you can use it for a specific task.

8051 MICROCONTROLLER

In 1981, Intel Corporation introduced an 8-bit microcontroller called the 8051. This microcontroller had 128 bytes of RAM, 4K bytes of on-chip ROM, two timers, one serial port, and four ports (each 8-bits wide) all on a single chip. The 8051 is an 8-bit processor, meaning that the CPU can work on only 8 bits of data at a time. Data larger than 8 bits has to broken into 8-bit pieces to be processed by the CPU. The 8051 has a total of four I/O ports, each 8 bits wide. Although the 8051 can have a maximum of 64K bytes of on-chip ROM, many manufacturers have put only 4K bytes on the chip. There are different flavors of the 8051 in terms of speed and amount of on-chip ROM, but they are all compatible with the original 8051 as far as the instructions are concerned. The various members of the 8051 family are 8051 microcontroller, 8052 microcontroller and 8031 microcontroller.

Block Diagram



Block diagram of inside the microcontroller 8051

8051 Microcontroller

The 8051 is the original member of the 8051 family. Figure 2.1 shows the block diagram of the 8051 microcontroller. The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set and pin out. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications. The AT89C51 provides the following standard

features: 4Kbytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bittimer/counters, five vector two-level interrupt architecture, a full duplex serial port, and on-chip oscillator and clock circuitry. In addition, the AT89C51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power-down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

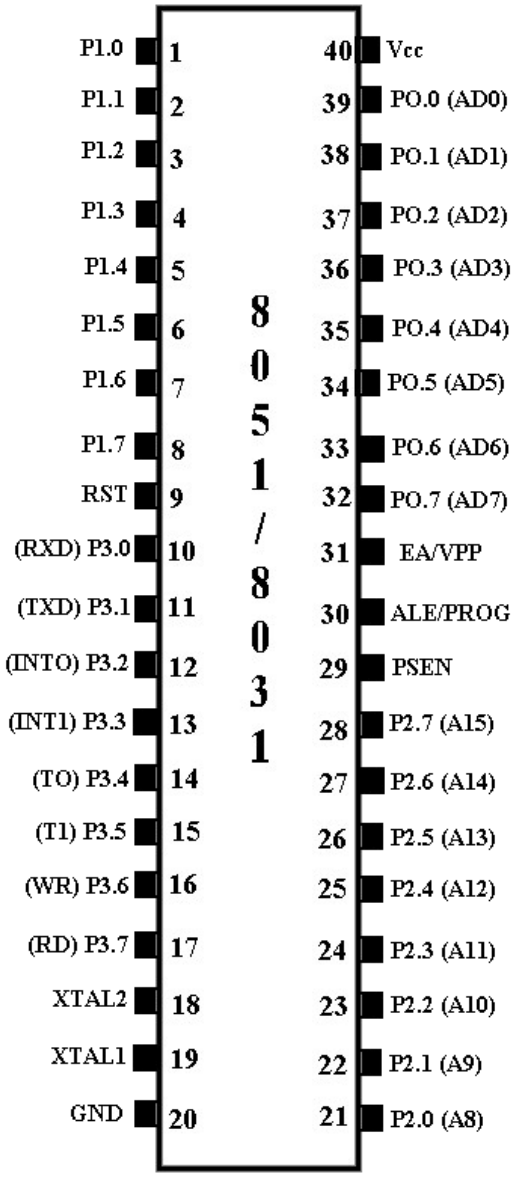
Pin Description

VCC

Supply voltage.

GND

Ground.



Pin diagram for microcontroller 8051

Port 0

Port 0 is an 8-bit open-drain bi-directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs. Port 0 may also be configured to be the multiplexed low- order address/data bus during accesses to external program and data memory. In this mode P0 has internal pull-ups. Port 0 also receives the code bytes during Flash programming, and outputs the code bytes during program verification. External pull-ups are required during program verification.

Port 1

Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. Port 1 also receives the low-order address bytes during Flash programming and verification.

Port 2

Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that uses 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that uses 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register. Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 3 output buffer scan sink/source four TTL inputs. When 1s are written to Port 3 pins they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL) because of the pull-ups. Port 3 also serves the functions of various special features of the AT89C51 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

Table 2.1function of port 3

Port 3 also receives some control signals for Flash programming and verification.

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/PROG

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming. In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory. If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

PSEN

Program Store Enable is the read strobe to external program memory.When the AT89C51 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

EA/VPP

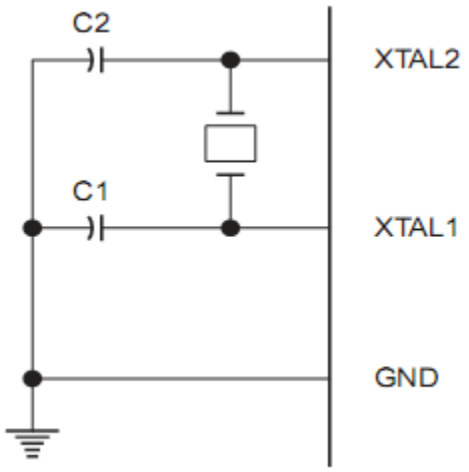
External Access Enable. EA must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, EA will be internally latched on reset. EA should be strapped to VCC for internal program executions. This pin also receives the 12-volt programming enable voltage (VPP) during Flash programming, for parts that require 12-volt VPP.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier. Oscillator Characteristics XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown.



Crystal Oscillator Connections

There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Programming of Microcontroller 8051

We are using embedded C programming language to program the central unit i.e. microcontroller 8051, so that it performs the specific task according to the requirement.

Need of C

Compiler produces hex file that we download into ROM of microcontroller. The size of hex file produced by compiler is one of the main concerns of microcontroller programmers for two reasons:

Microcontroller has limited on -chip ROM

The code space for 8051 is limited to 64 KB

Programming in assembly language is tedious and time consuming. C is a high level programming language that is portable across many hardware architectures.

So for following reasons we use C

It is easier and less time consuming to write in C than assembly.

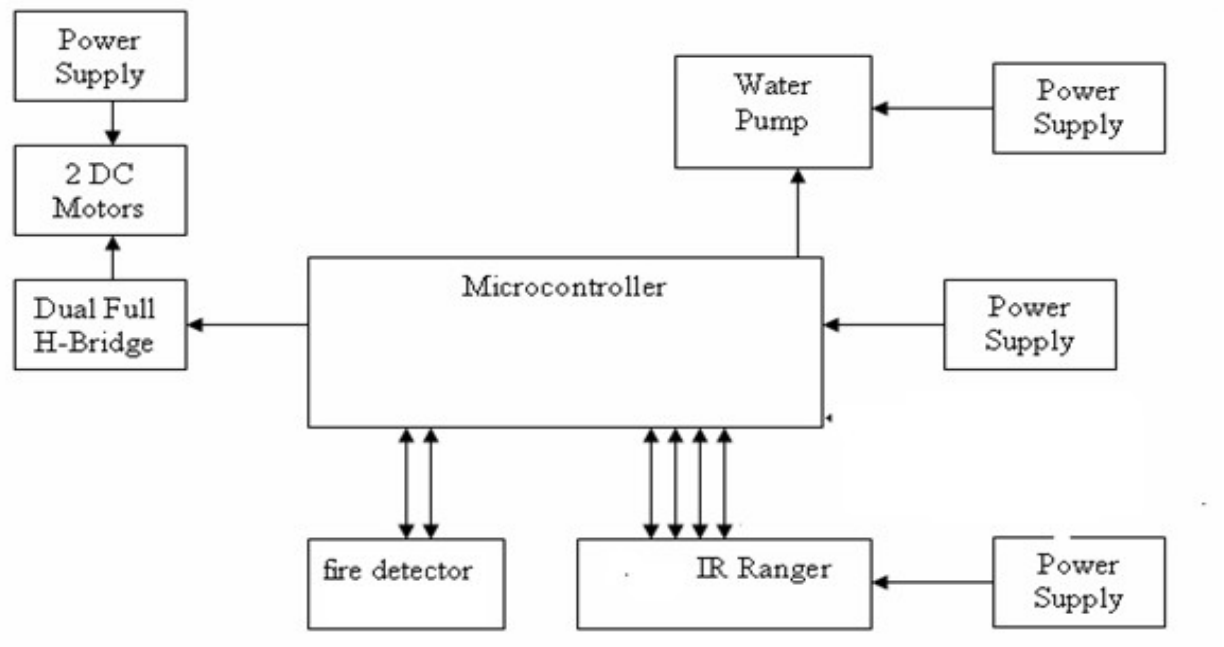
C is easier to modify and update.

You can use code available in function libraries.

C code is portable to other microcontrollers with little or no modification.

We use reg51.h as a header file as "#include <reg51.h>". These files contain all the definitions of the 80C51 registers. This file is included in your project and will be assembled together with the compiled output of your C program.

Block Diagram

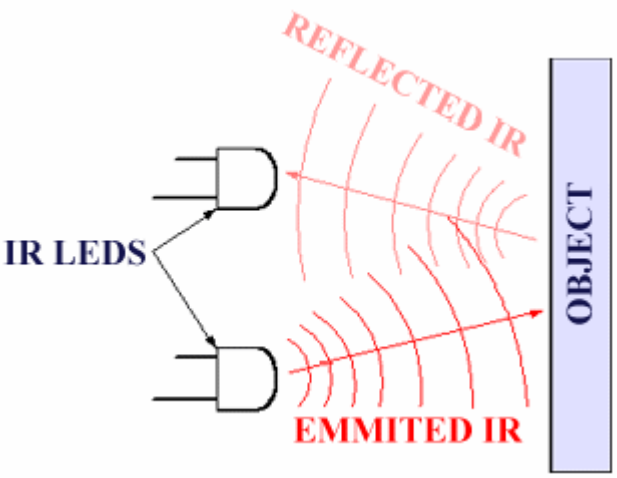


Sensors part:-

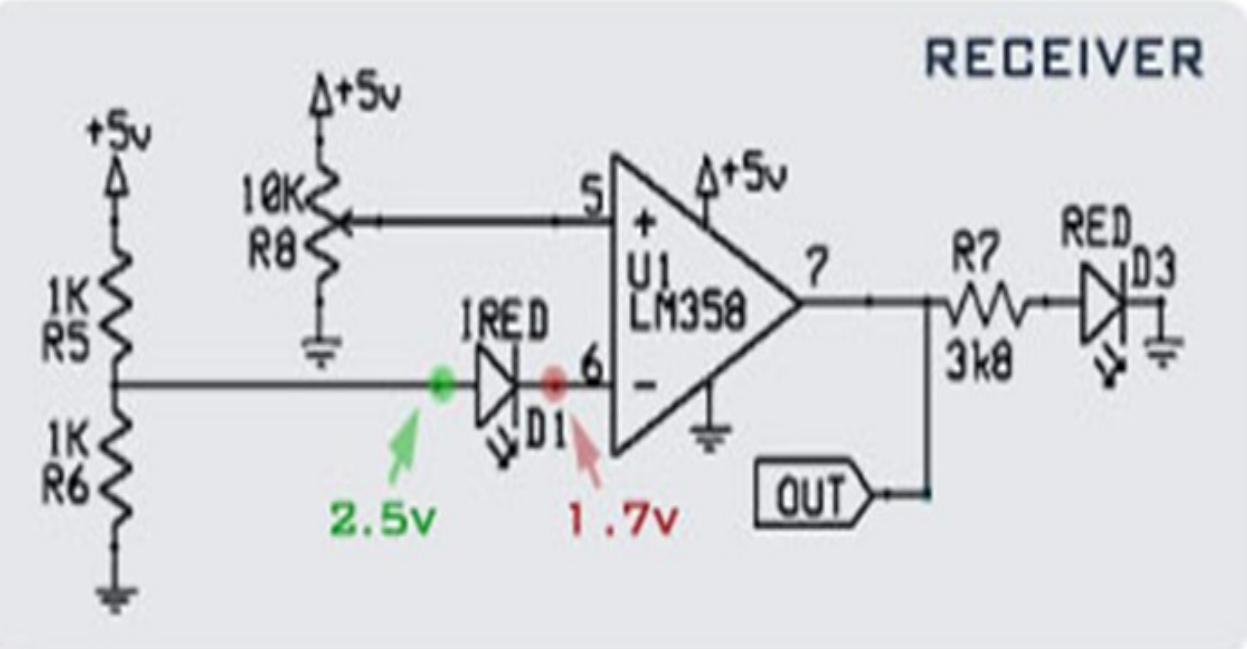
The IR Sensor

It is the same principle in ALL Infra-Red sensors. The basic idea is to send infra red light through IR-LEDs, which is then reflected by any object in front of the sensor.

Then all you have to do is to pick-up the reflected IR light. For detecting the reflected IR light, we are going to use a very original technique: we are going to use another IR-LED, to detect the IR light that was emitted from another led of the exact same type!

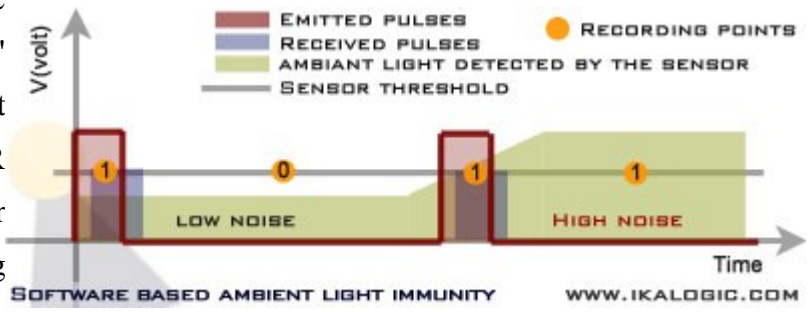


This is an electrical property of Light Emitting Diodes (LEDs) which is the fact that a led Produce a voltage difference across its leads when it is subjected to light. As if it was a photo-cell, but with much lower output current. In other words, the voltage generated by the leds can't be - in any way - used to generate electrical power from light, It can barely be detected. that's why as you will notice in the schematic, we are going to use a Op-Amp (operational Amplifier) to accurately detect very small voltage changes.



When the sensor is controlled by a microcontroller to generate the low duty cycle pulses, you can benefit from the High and Low pulses to be able to detect any false readings due to ambient light. This is done by recording 2 different outputs of the sensor, one of them during the ON pulse (the sensor is emitting infra red light) and the other during the OFF time. and compare the results.

The Idea is enlightened by this graph, where in the first period, there is low ambient noise, so the microcontroller records a "1" during the on cycle, meaning that an object reflected the emitted IR Light, and then the microcontroller records a "0" meaning that during the OFF time, it didn't receive anything, which is logic because the emitter LED was OFF. But study the second period of the graph, where the sensor is put in a high ambient light environment. As you can see, the the microcontroller records "1" in both conditions (OFF or ON). This means that we can't be sure whether the sensor reception was caused by an object that reflected the sent IR light, or it is simply receiving too much ambient light, and is giving "1" whether there is an obstacle or not.



The following table show the possible outcomes of this method.

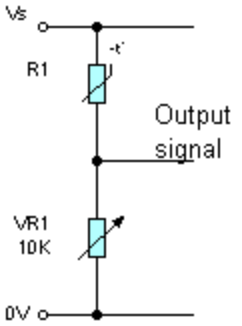
Output recorded during:		Software based deduction
On pluse	Off time	
1	0	There is definitely an Obstacle in front of the sensor
1	1	The sensor is saturated by ambient light, thus we can't know if there is an obstacle
0	0	There is definitely Nothing in front of the sensor, the way is clear
0	1	This reading is un logical, there is something wrong with the sensor.

FIRE SENSOR(thermistor):-

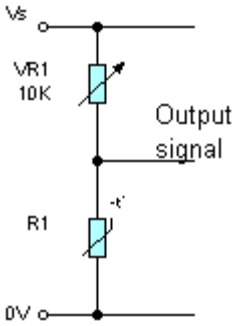
How does it operate?

Usually the temperature sensor produces a voltage signal that increases as the temperature increases. The inverted temperature sensor (cold sensor) produces a voltage signal that increases as the temperature decreases.

If the temperature sensor is being used with a digital process unit then it needs to be followed by a comparator or Schmitt inverter to give a sharp change of signal from low to high.

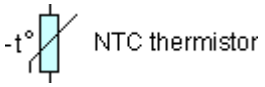


Normal circuit



Inverted circuit

The temperature sensing circuit uses an NTC (negative-temperature coefficient) thermistor to monitor temperature.

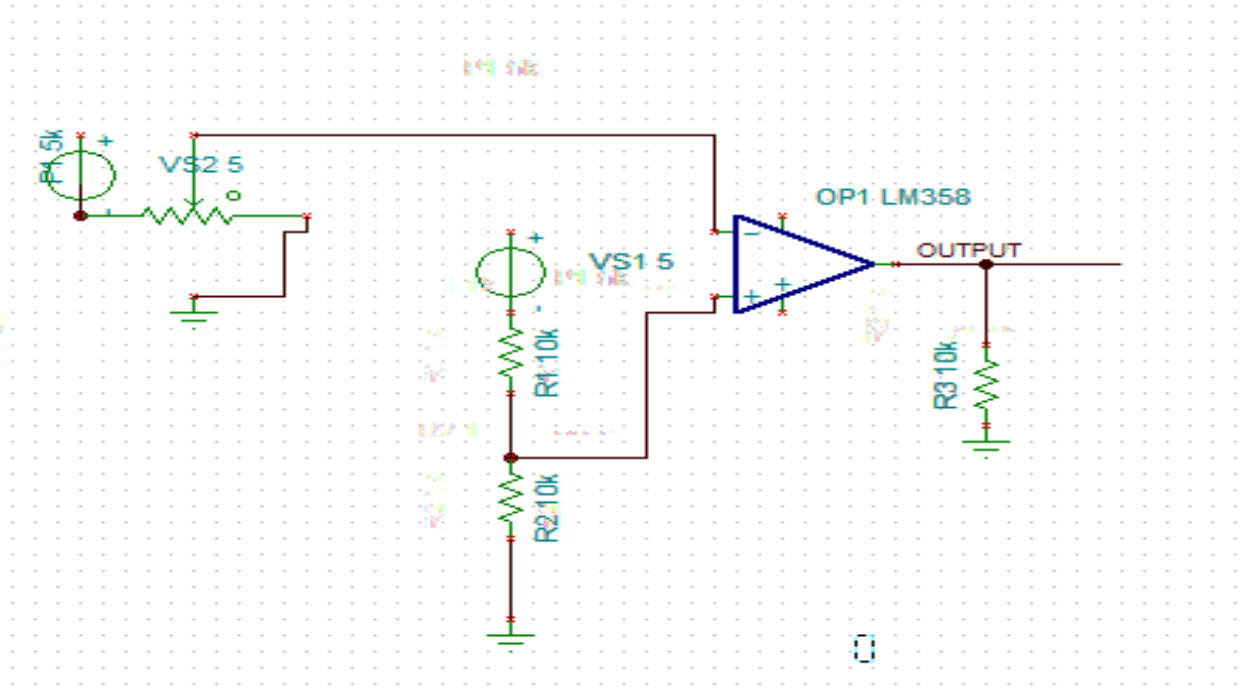


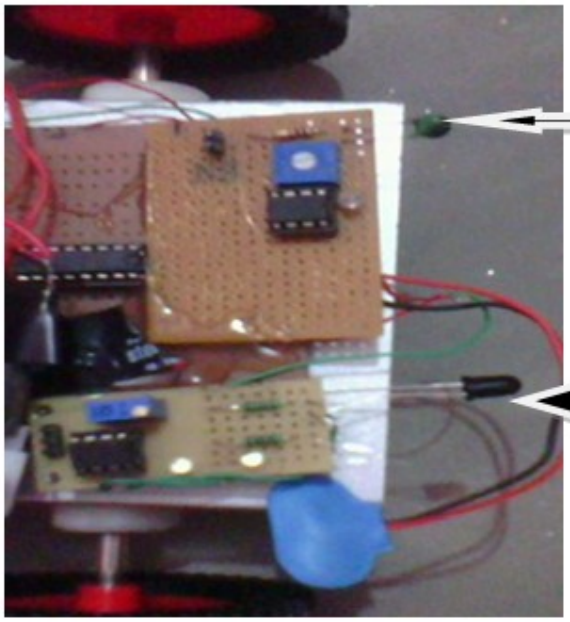
The resistance of a NTC thermistor falls as its temperature increases.

Click on the circuit diagram to download a Livewire file of the circuit that you can investigate and add to your own circuit.

The temperature sensor circuit is a potential divider whose output voltage is determined by the upper and lower parts of the circuit.

In the normal circuit, the thermistor is placed in the upper half of the potential divider. In the inverted circuit, the thermistor is placed in the lower half of the potential divider.





FIRE SENSOR(thermistor)

The IR Sensor

8. DC GEAR MOTOR

The two types of motor's that you are likely to use in robotic adventure are DC motors and RC servo motors. The most common motor for robotics is the DC gear motor, which works by gearing down a fast Dc motor to make the motor turn at a slower speed and give the motor a higher torque suitable for robot locomotion.

A dc gear motor is basically a regular DC motor with a special gear box attached to the output shaft . Your robot electrical drive circuitary can control the dc gear motor to rotate the wheels of your robot for locomotion.

You can get a DC motor without a gear head, but generally these are too fast (around 15,000 RPM). For a robot to move at a reasonable rate you have to gear down a DC motor to about 30 to 80 RPM . When you gear down a DC motor, you get a slower speed and plenty of torque.

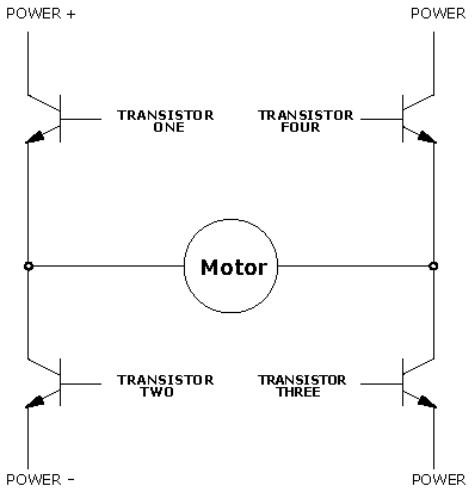
H BRIDGE CIRCUIT.

One of the most popular motor controller circuits is an H bridge circuit. An H bridge circuit turns a motor on and off, allows a computer or processor to control a motor's direction and regulate speed, and may even provide a breaking mechanism . A dc gear motor's rotation direction is usually controlled with an H bridge circuit.

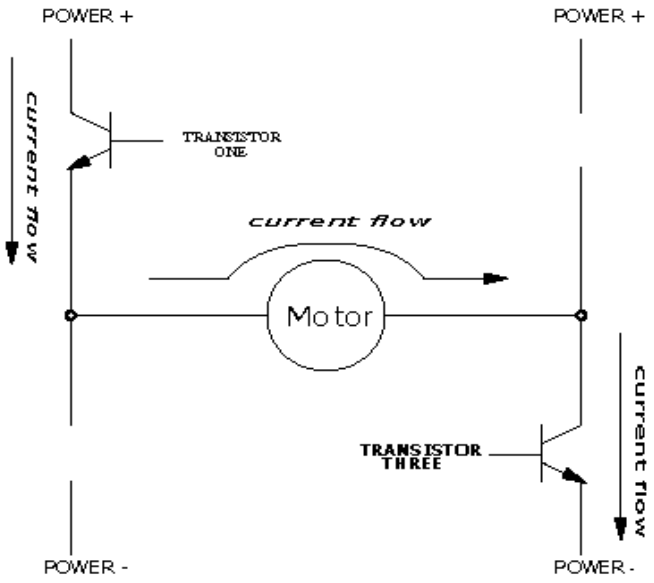
A processor can not control a motor directly for a several reasons. First, a computer doesnot output enough power to drive a motor. Second, a computer cannot control direction because it has only outputs. Third, motors are noisy electrically speaking, and would quickly damage a computer . essentially, the computer sends, signals to the H bridge to tell it to go forward, reverse, brake or add speed.

The H bridge then steps up the voltage and power for the motor. The H bridge circuits also isolates the computer or processor from destrutive voltage spikes and noise, which arise maily from motors. In addition to using an H bridge circuit, you might want to have two sets of batteries: one for your electronics and another for your motors.

The H-Bridge Circuit



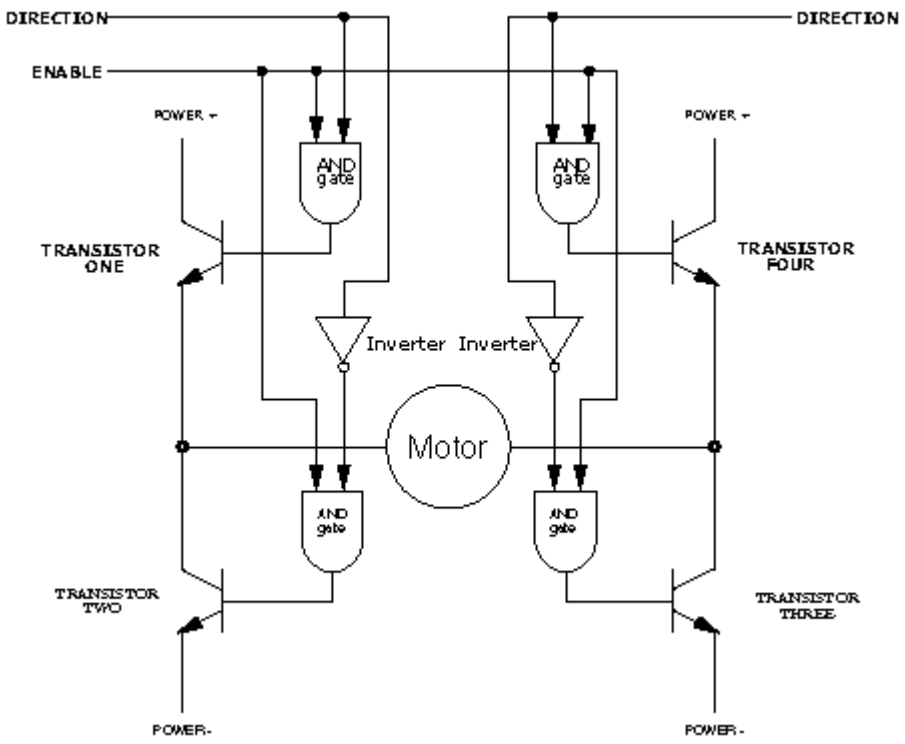
This circuit known as the H-bridge (named for its topological similarity to the letter "H") is commonly used to drive motors. In this circuit two of four transistors are selectively enabled to control current flow through a motor.



opposite pair of transistors (Transistor One and Transistor Three) is enabled, allowing current to flow through the motor. The other pair is disabled, and can be thought of as out of the circuit.

By determining which pair of transistors is enabled, current can be made to flow in either of the two directions through the motor. Because permanent-magnet motors reverse their direction of turn when the current flow is reversed, this circuit allows bidirectional control of the motor.

The H-Bridge with Enable Circuitry



It should be clear that one would never want to enable Transistors One and Two or Transistors Three and Four simultaneously. This would cause current to flow from Power + to Power - through the transistors, and not the motors, at the maximum current-handling capacity of either the power supply or the transistors. This usually results in failure of the H-Bridge. To prevent the possibility of this failure, enable circuitry as depicted in Figure is typically used.

In this circuit, the internal inverters ensure that the vertical pairs of transistors are never enabled simultaneously. The Enable input determines whether or not the whole circuit is operational. If this input is false, then none of the transistors are enabled, and the motor is free to coast to a stop.

By turning on the Enable input and controlling the two Direction inputs, the motor can be made to turn in either direction.

Note that if both direction inputs are the same state (either true or false) and the circuit is enabled, both terminals will be brought to the same voltage (Power + or Power - , respectively). This operation will actively brake the motor, due to a property of motors known as back emf, in which a motor that is turning generates a voltage counter to its rotation. When both terminals of the motor are brought to the same electrical potential, the back emf causes resistance to the motor's rotation.

Software Design

Hardware Drivers

Specifications

The hardware drivers are the low-level portions of the robot's code. The purpose of these drivers is to provide a hardware to software interface that enables access to such hardware as the sensor and motors from within the behavioural code. The specifications for these functions were determined by the software development team before the coding process started. This was to ensure that the code would be compatible between the upper and lower layers of soft-ware. Only two main hardware drivers were necessary for this project. In addition to these drivers several internal routines were written to handle such things as timer interrupts and MCU initialization.

This function is the low-level driver that communicates with the dis-tance sensors and loads global variables with the returned results. This driver is responsible for initializing the sensors for each reading, taking the readings, processing them, and then loading them for use by the higher level software.

It had been our understanding that the IR Sensors returned a binary value that represented a distance. We also expected that this relationship would be linear. 1. Tabulate the relationship for a range of measured distances

2. Graph this relationship

3. Use mathematical software to model the graph as an equation

4. Put this equation and its parameters into the code and perform the conversion for each of the three sensors

Mechanical

Chassis Specifications

Firstly we needed to analyse what the robot needed to achieve and how we were to design and build a chassis to meet those requirements. We decided to keep the robot chassis as simple as possible to achieve the tasks it needed to. We discussed the specifications of the robot and came to a number of

conclusions. The robot needed to be: light in weight, small and an easy shape to move around the table and avoid obstacles and cheap to manufacture.

Initial concepts

We had two initial ideas for the shape of the chassis. The first was a rectangular platform with two drive wheels and a pivot bearing at the rear.

SOFTWARE DETAILS

Software used in our project is keilc. Its details are as follows:

Keil (IDE) MicroVision3

Keil Software development tools are used to create products for practically every industry: consumer electronics, industrial control, networking, office automation, automotive, space exploration. Micro Vision Two is a second generation IDE that simplifies project development and application testing. With Micro Vision Two, we can easily create embedded applications in a mixture of C and assembly. Real-time applications benefit from our highly optimized C libraries and real-time kernels.

MicroVision3 provides a centralized front-end interface for the compiler, assembler, linker, debugger, and other development tools. The Project Window in MicroVision3 displays the current target, groups, and source files that comprise our project. Rather than creating a single target for each project, MicroVision2allows multiple targets for each project file. So, with a single project file, we can create a target for simulating, a target for our emulator, and a production target for programming into EPROM {E-PROM}.

Each target is composed of one or more groups which are in turn composed of one or more source files. Groups let us divide the source files into functional blocks or assign source files to different team members. Options may be configured at each level of the project. This gives us a great deal of freedom and flexibility when organizing our application. In addition to the on-line help, MicroVision3 provides on-line versions of the development tool manuals as well as the device manuals.

Keil C Compilers are based on the ANSI standard and include extensions necessary to support the 8051, 251, and 166 microcontroller families. The optimizer in our compiler is tuned for each specific architecture and provides the highest level of code density and execution speed.

The Keil C compilers give full us control over our embedded platform. We decide which register banks are used, when to access certain memory areas, which variables are stored in bits, when and how to use special function registers, and so on. Without ever writing any assembly code we may even write interrupt service routines in C. Code generated by the Keil C Compiler compares with that of a professional assembly programmer. This is due to the level of optimizations that are performed. One such optimization is global register optimization.

By analyzing which registers are used in each function, the compiler can better optimize register usage program-wide and generate smaller, faster programs. This is accomplished by iterative compilation steps during the make process.

The MicroVision3 debugger is designed to make testing your programs as efficient as possible. While editing and debugging your programs, text and code attributes are displayed in the source window. As you step through your program, the current line is marked with a yellow arrow. Code coverage shows you which lines of your program have been executed. Green means the line has been run. Grey means is has not.

Breakpoints are clearly marked in the source window. Red for enabled, white for disabled. These attributes make following program flow easier than ever. The features of the Micro Vision Two debugger don't stop there. When simulating your programs, you not only get source-level, symbolic simulation. You also get on-chip peripheral simulation. Dialog boxes display the condition of all peripherals and on-chip components.

Source Code

```
org 0000h
mov p1,#ffh          //i/p port
mov p2,#00h          // o/p port

back:mov a,p1
    cjne a,#02h,l1
acall delay
mov p2,#05h
acall delay
sjmp back
```

```
l1:cjne a,#04h,l2
acall delay
mov p2,#08h
acall delay
sjmp back

l2: cjne a,#08h,l3
acall delay
mov p2,#0ah
acall delay
sjmp back

l3: cjne a,#05h,l4
acall delay
mov p2,#00h
acall delay
sjmp back

l4: cjne a,#06h,l5
acall delay
mov p2,#01h
acall delay
sjmp back
l5: nop

sjmp back

delay: mov r2,#255
again:mov r1,#50
here:djnz r1,here
        djnz r2,again
          ret
```

end

SIMULATION

Simulation of our project is done with help of Proteus. Its details are as follows:

PROTEUS

Proteus 7.4 SP3 Portable is a software technology that allows creating clinical executable decision support guidelines with little effort.

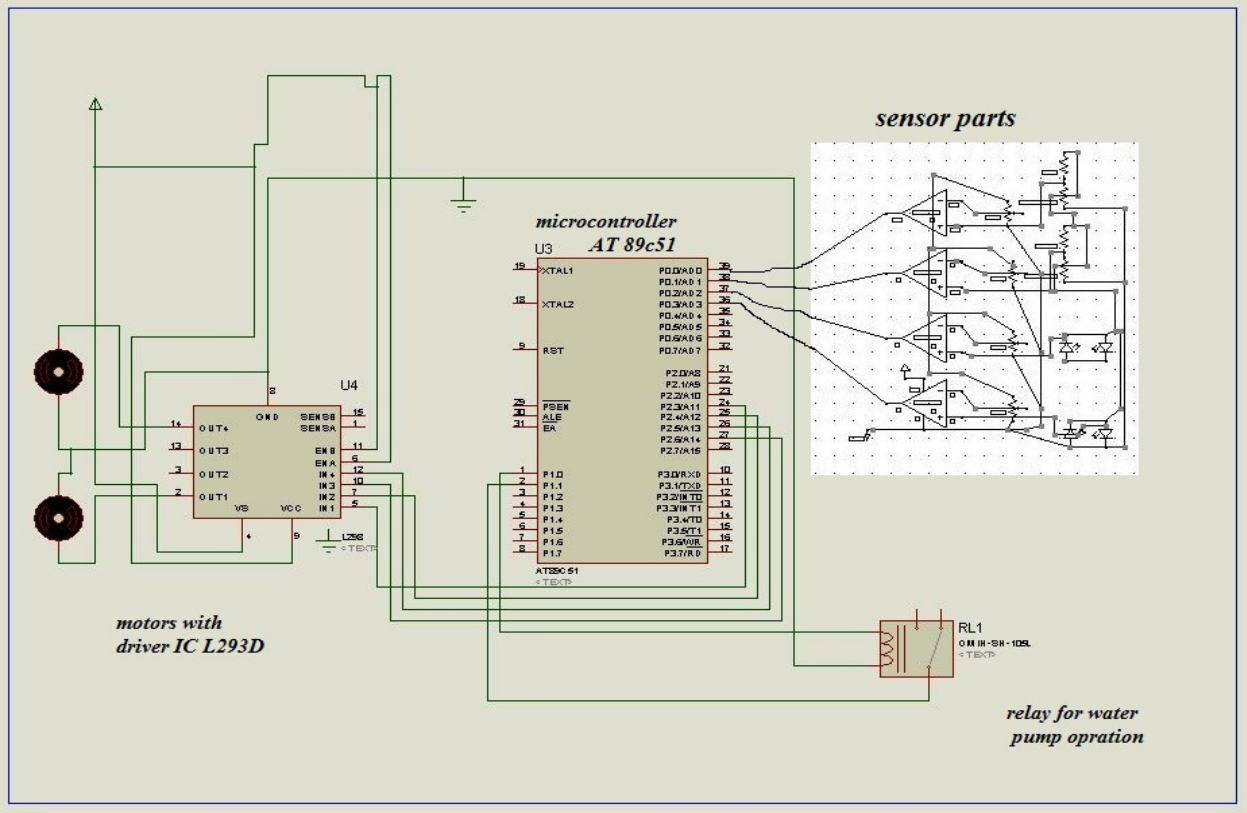
A software tool that allows creating and executing clinical decision support guidelines using the Proteus approach is available. The tool called Protean may be downloaded from here. Protean allows creating new guidelines or editing existing ones very easily. Much of the editing is done by dragging and dropping.

The Proteus guidelines are created with modular entities called Knowledge Components (KCs). Each KC represents a clinical activity and is available to the clinician as a module of executable knowledge with its own intelligence.

Experts at remote locations may manage individual KCs, keeping them in sync with the current medical concepts, while the clinicians automatically get the state-of-the-art executable knowledge. This is akin to opening a web page using a hyperlink; the user gets the fresh content by clicking on the same URL when the author of the web page updates it. Unlike a web page however, the Proteus KCs are executable knowledge and not passive information. Each guideline may have many KCs, each being updated by a different expert or a group of experts.

The intelligent decision-making in the KC comes from the Inference Tools in the Proteus approach. Anything that can make the inferences that a KC needs can be declared its inference tool. Simple software algorithms, sophisticated artificial intelligence tools or even remote human experts can be specified as inference tools for KCs. The inference tool can be as easily swapped as they can be declared. Therefore, if a tool with better inferencing capabilities becomes available, it can be used to replace the previous one in a few simple steps.

SIMULATION


sensor parts
microcontroller
AT 89c51
U3
U4
motors with
driver IC L293D
RL1
relay for water
pump opration

COMPONENT COST

Components	Quantity	Price
MICROCONTROLLER 8051	1	70
PCB	1	120
BASE-40 PIN	1	6
DC GEAR MOTOR	2	300
IR SENSOR	2	35
ULN2803	1	15
8-PIN BASE	2	6
LED	2	2
THERMISTER	2	20
CASTER Wheel	1	25
WHEELS	2	60
PUMP	1	150
CHESIS	1	70
10K	4	4
RELAY(12V)	1	10
7805	1	7
11.059MHZ	1	5
LM358	2	10
20k pot	2	10
33pf	2	4
MALE CONNECTOR	1	5
FEMALE CONNECTOR	1	5
L293D	1	130

MOTOR With FAN	1	75
BATTERY	1	15
BATTERY CAP	1	5

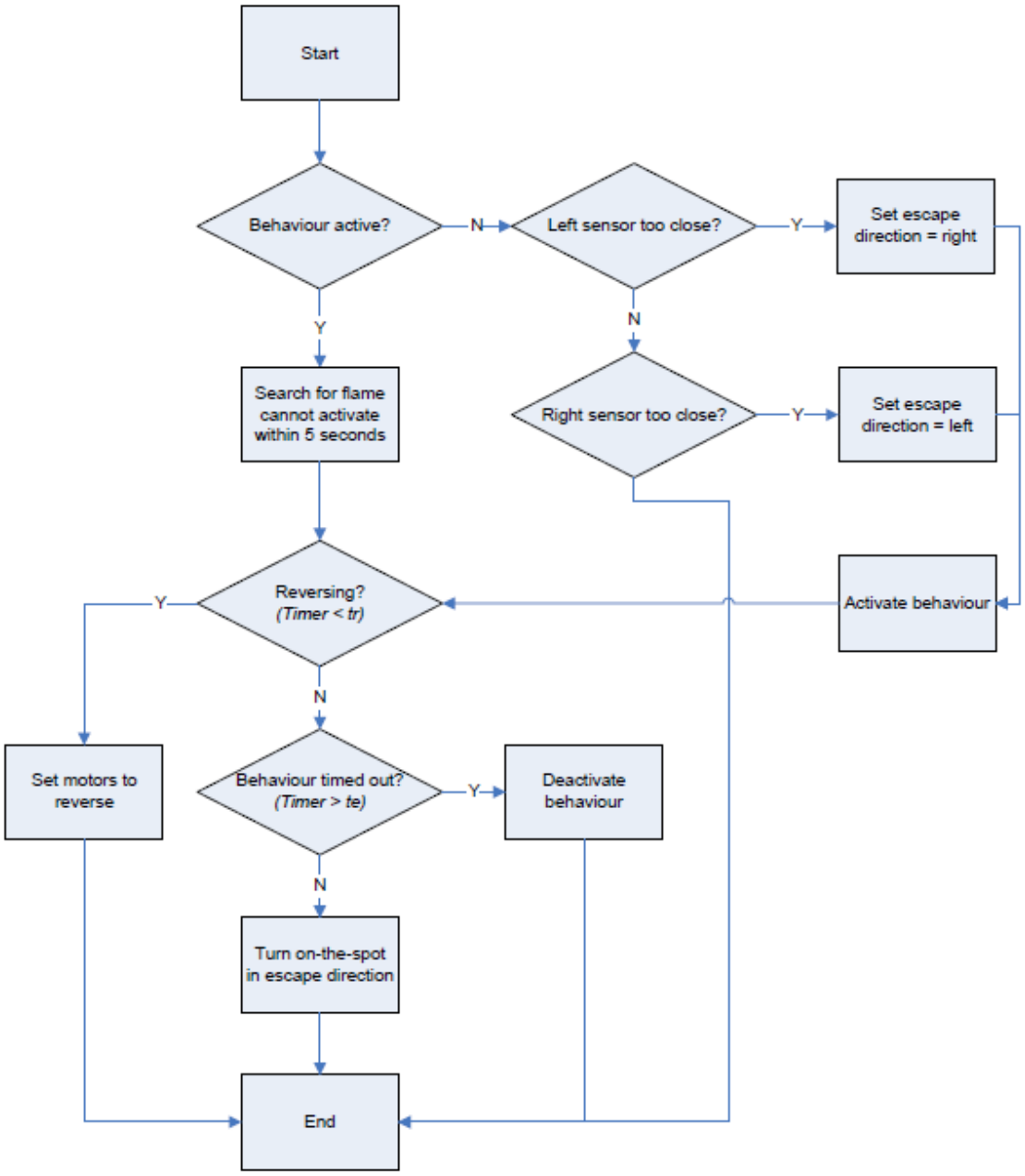
Flowcharts

Escape

The aim of the escape behaviour is to prevent the robot from colliding with obstacles when avoid fails.

Escape works by taking the distances from both sensors; if one is under a safety threshold, the behaviour takes over. First it reverses the robot for a short period of time; it then turns slightly less than 90 degrees away from the sensor which was closest (an object on the right gets too close, the robot will turn left)

If this behaviour becomes active it will prevent the “Search for flame” behaviour from activating for another 5 seconds.



Follow Flame

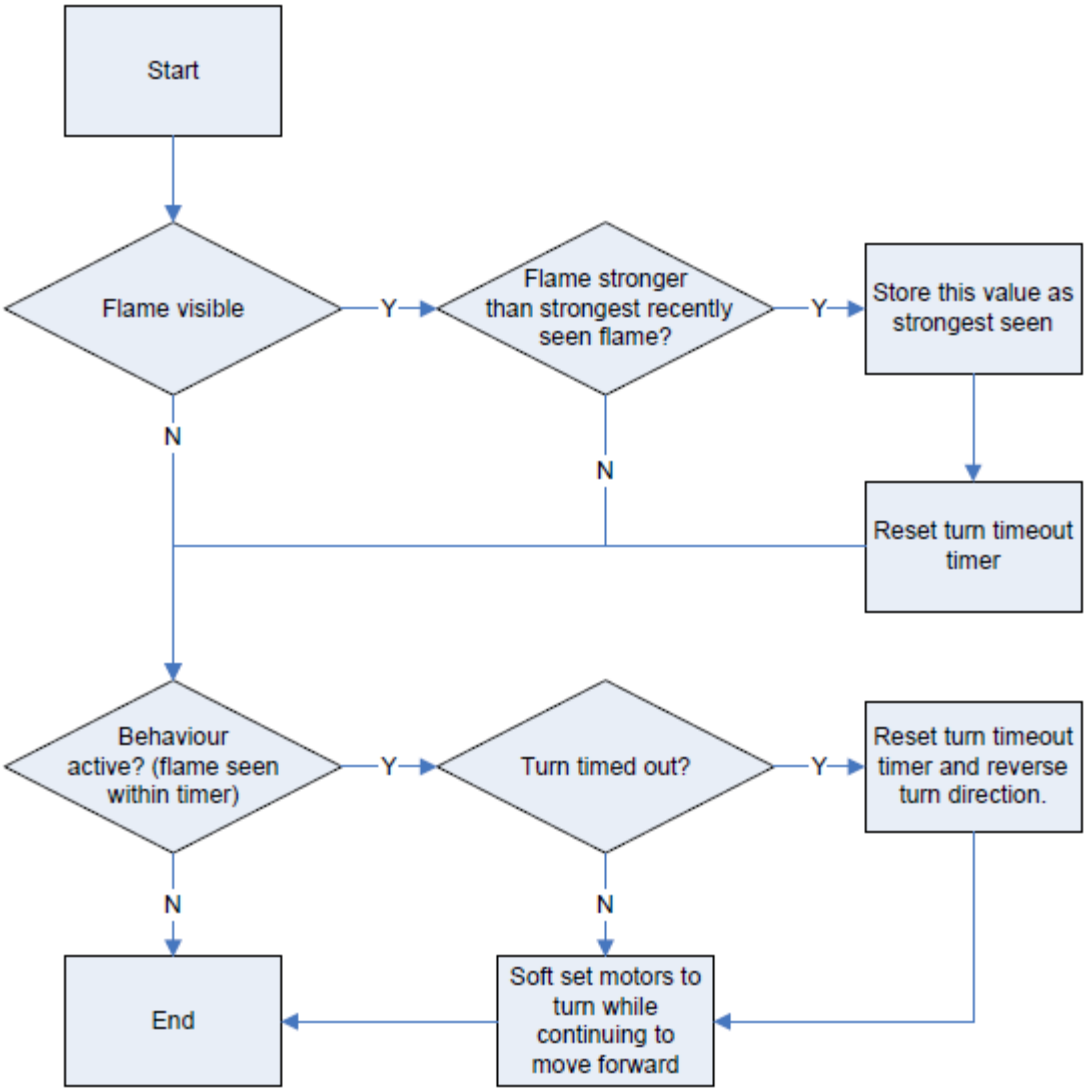
This behaviour will guide the robot to a flame so it can be extinguished.

Due to the fact that our flame detector could only give a single output as to whether there was a flame in front of it or not this task became more difficult.

To track the flame we needed to know where it was in front of the robot, so to do this we made a scan like motion, whereby the robot would see a flame, turn away from it for a given time after it

last saw the flame, then reverse the scan, seeing the flame again and continue to turn for a given amount of time after the flame dropped out of view, then turned back, and so on.

The sensor did perform (although very inaccurately) the function of reporting the distance to the flame. From this we were able to say we are close enough to the flame and put it out, which we did by turning on the fan (which then triggered the "Extinguish flame" behaviour.)



Testing

Initial testing

The programming of the control system was a continuous process; this involved implementing a new component, as few were introduced at a time as possible. Often we would find that the new component wouldn't act as expected. This would lead to the need to comment out the new control code and go though it line by line, making sure that it was not causing the problem.

Along with the testing of new components we needed to test individual behaviours. Due to the nature of a behaviour based system it was relatively easy to comment out existing and fully functional behaviours, leaving only the one which was in development. This made it possible for us to identify issues that we would probably have missed if trying to complete the entire project at once.

One of the major things we found was that if a candle was too far from the robot to be extinguished yet the fire sensir could still see it the robot would continue to attempt to put it out. To solve this we added a maximum number of times the fan could be turned on without having a break of some time in between. This meant that if the robot attempted to put out the same flame more than four times the fan behaviour would be disabled; this in turn would allow follow flame to take over and bring the robot closer, until the time when the extinguishing behaviour was re-enabled.

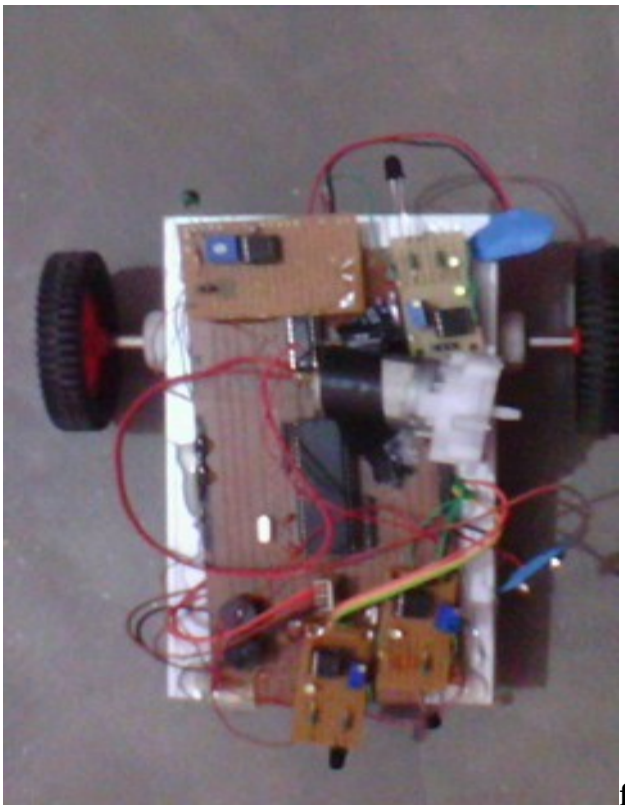


fig:- complete picture of robot.

Problems Encountered

Drive Motors design :- There were numerous problems with the motor and gearbox packages we used for the Fire-Fighting Robot. The motors provided sufficient driving torque and speed but where our problems started was attaching the motors to the wheels.We could not drill the shaft and insert a grub screw so we opted to glue the motor shafts to the wheels.

Mounting of sensors :- difficult to align ir sensor and fire sensor in front of the flame.

Flame sensor:- not able to hone in on the flame .

. Conclusion

Uses sensors to pinpoint exact location of candle.The robot accurately and efficiently finds the fire within the allotted time after the fire alarm is heard and returns to a safe place (Home).
The chassis combined with the centrally located wheels worked extremely well for the robot and its ability to navigate the world in which it was placed. The FIRE SENSOR (thermistor) created some difficulty in that it was not originally direction sensitive, and thus required the addition of a custom designed shell. It was extremely reliable once we had completed this physical modification

Bibliography

These are some of the sites we visited:-

http://www.hobbyengineering.com/rmapIndex.html
http://www.ridgesoft.com/buildingbots.htm
http://www.seattlerobotics.org/guide/infrared.html
http://www.mstracey.btinternet.co.uk/interest.htm
http://www.dudi30.republika.pl/galeria/index.html //New robot every week
http://www.robotics.com/robomenu/index.html