Shortest Path Finder

Project Advisor:
Mr. Asmar Azar Khan

Group Members:

RAO ZAHOOR AHMAD	050620-012
MUHAMMAD TAHIR	050620-101
QAMAR HASSAN IQBAL	050620-142

Bs- Telecommunication Engineering

[School of Science and Technology]
UNIVERSITY OF MANAGEMENT AND TECHNOLOGY, LAHORE (PAKISTAN)

Statement of Submission

Shortest Path Finder

Documentation of Final Project submitted to the Faculty of School of Science and Technology, University of Management and Technology, Lahore (Pakistan).

In Partial Fulfillment of the requirements for the degree of
BACHELOR OF SCIENCE (H)
IN
TELECOMMUNICATION ENGINEERING

_____	_____
Project Advisor	Project Coordinator

ACKNOWLEDGEMENTS

All the deepest sense of gratitude to almighty ALLAH, the most compassionate and merciful who is omnipotent and omnipresent, and has divulged His knowledge to man.

We are highly indebted to our advisor Mr. Asmar Azar Khan, who made us work and think hard to collaborate and improvise for the successful completion of the assigned task.

Heartiest thanks to Mr. Hafiz Tanveer for his continued cooperation, support and tireless help towards the successful completion of this project.

- RAO ZAHOOR AHMAD 050620-012
- MUHAMMAD TAHIR 050620-101
- QAMAR HASSAN IQBAL 050620-142

ABSTRACT

The goal our project is to develop a Shortest Path Finder (SPF) with specified point on ground for indoor environment. Calculating the shortest path with the help of Dijkstra's Algorithm and moves the vehicle at given path from the Dijkstra's algorithm. As a Engineering Student we get opportunity to teach the Vehicle how to follow the line thus giving it a human-like property of responding to Stimuli.

It's automatically changes the direction of the vehicle without external command or guidance.

In this report we are also adding our Final project documentation.

Table of Contents

Chapter # 1

Introduction

1 Introduction

1.1 Introduction:

We have to design the Autonomous Vehicles that search the shortest path from one point to the other point follow their path to go start point to the end point. Autonomous vehicles are already in use in many manufacturing facilities, and they are also sold as toys. The goal our project is to develop a Shortest path finder Vehicles that can follow a path specified and starting and ending point is given on ground for indoor environment, Vehicles Calculating the shortest path between the nodes and moving from starting point to the ending. As an engineering student we get an opportunity to "teach" the vehicles how to follow the shortest path thus giving it a human-like property of responding to stimuli.

Webster dictionary define robot as "an automatic device that performs function ordinarily ascribed to human beings " with the definition we say our Vehicle is a robot device that can be perform the work with out any external instruction.

Perhaps one day, the world's highways will be filled with cars requiring on user intervention. While we are not there yet, and may never get there, an autonomous vehicle can serve as a very interesting and practical design exercises.

Our vehicle uses the Dijkstra's algorithm to follow a path on the floor to reach from starting point to the ending point. It can automatically changes its direction without external commands or guidance, can carry different types of equipment. Some autonomous vehicles used in manufacturing use as a similar concept, following magnetic strips embedded in the factory floor. The versatility of an optical design should be readily apparent, since re-routing the vehicle requires no more then re-printing.

1.2 Scope:

In recent years, the technology is progressing so fast that almost every field is getting inspired from it. The work in field of automating the vehicle has been in progress for last many years. The military is especially interested in the automated vehicles either is ground, or aerial or naval. They are ambitions to replace today's Abram's tanks and Bradley Fighting Vehicles with swarms of unmanned, autonomous vehicles bristling with missiles and long-range guns. Moreover many progresses are made in field of vision in autonomous vehicles. These are reducing the rate of accidents on road in routine life and help the people to drive more safely. In the field of agriculture, the autonomous ground tractors are developed for carriage and performing other task.

1.3 Project Application:

- Carriage of goods in industries.
- Military purpose.
- Mining purpose.
- Measurements of dimensions of roads and building.
- Traffic information systems use Dijkstra's algorithm in order to track the source and destinations from a given particular source and destination.

- OSPF- Open Shortest Path First, used in Internet routing.

1.4 Major Component of Vehicle:

Although the mechanical, electrical and computational structure of vehicle can vary considerably, most have the following major components in common.

- A manipulator(the mechanical unit)
- Controller(the brain)
- Power Supply

Chapter # 2
Microcontroller

2 Microcontroller

2.1 Introduction:

A microcontroller (also microcontroller unit, MCU or µC) is a small computer on a single integrated circuit consisting of a relatively simple CPU combined with support functions such as a crystal oscillator, timers, watchdog, serial and analog I/O etc. Program memory in the form of NOR flash or OTP ROM is also often included on chip, as well as a, typically small, read/write memory.

Microcontrollers are designed for small or dedicated applications. Thus, in contrast to the microprocessors used in personal computers and other high-performance or general purpose applications, simplicity is emphasized. Some microcontrollers may operate at clock frequencies as low as 32kHz, as this is adequate for many typical applications, enabling low power consumption (mill watts or microwatts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a Digital signal processor (DSP), using higher clock speeds and not needing such very low powered operation.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, remote controls, office machines, appliances, power tools, and toys. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

2.2 Embedded design:

The majority of computer systems in use today are embedded in other machinery, such as automobiles, telephones, appliances, and peripherals for computer systems. These are called embedded systems. While some embedded systems are very sophisticated, many have minimal requirements for memory and program length, with no operating system, and low software complexity. Typical input and output devices include switches, relays, solenoids, LEDs, small or custom LCD displays, radio frequency devices, and sensors for data such as temperature, humidity, light level etc. Embedded systems usually have no keyboard, screen, disks, printers, or other recognizable I/O devices of a personal computer, and may lack human interaction devices of any kind.

2.3 Interrupts:

Microcontrollers must provide real time (predictable, though not necessarily fast) response to events in the embedded system they are controlling. When certain events occur,

an interrupt system can signal the processor to suspend processing the current instruction sequence and to begin an interrupt service routine (ISR, or "interrupt handler"). The ISR will perform any processing required based on the source of the interrupt before returning to the original instruction sequence. Possible interrupt sources are device dependent, and often include events such as an internal timer overflow, completing an analog to digital conversion, a logic level change on an input such as from a button being pressed, and data received on a communication link. Where power consumption is important as in battery operated devices, interrupts may also wake a microcontroller from a low power sleep state where the processor is halted until required to do something by a peripheral event.

2.4 Programs:

Microcontroller programs must fit in the available on-chip program memory, since it would be costly to provide a system with external, expandable, memory. Compilers and assembly language are used to turn high-level language programs into a compact machine code for storage in the microcontroller's memory. Depending on the device, the program memory may be permanent, read-only memory that can only be programmed at the factory, or program memory may be field-alterable flash or erasable read-only memory.

2.5 Higher integration:

In contrast to general-purpose CPUs, microcontrollers may not implement an external address or data bus as they integrate RAM and non-volatile memory on the same chip as the CPU. Using fewer pins, the chip can be placed in a much smaller, cheaper package.

Integrating the memory and other peripherals on a single chip and testing them as a unit increases the cost of that chip, but often results in decreased net cost of the embedded system as a whole. Even if the cost of a CPU that has integrated peripherals is slightly more than the cost of a CPU + external peripherals, having fewer chips typically allows a smaller and cheaper circuit board, and reduces the labor required to assemble and test the circuit board.

A microcontroller is a single integrated circuit, commonly with the following features:

- central processing unit - ranging from small and simple 4-bit processors to complex 32- or 64-bit processors
- discrete input and output bits, allowing control or detection of the logic state of an individual package pin
- serial input/output such as serial ports (UARTs)
- other serial communications interfaces like I²C, Serial Peripheral Interface and Controller Area Network for system interconnect
- peripherals such as timers, event counters, PWM generators, and watchdog
- volatile memory (RAM) for data storage
- ROM, EPROM, EEPROM or Flash memory for program and operating parameter storage
- clock generator - often an oscillator for a quartz timing crystal, resonator or RC circuit
- many include analog-to-digital converters
- in-circuit programming and debugging support

This integration drastically reduces the number of chips and the amount of wiring and circuit board space that would be needed to produce equivalent systems using separate chips. Furthermore, and on low pin count devices in particular, each pin may interface to several internal peripherals, with the pin function selected by software. This allows a part to be used in a wider variety of applications than if pins had dedicated functions. Microcontrollers have proved to be highly popular in embedded systems since their introduction in the 1970s.

Some microcontrollers use a Harvard architecture: separate memory buses for instructions and data, allowing accesses to take place concurrently. Where a Harvard architecture is used, instruction words for the processor may be a different bit size than the length of internal memory and registers; for example: 12-bit instructions used with 8-bit data registers.

The decision of which peripheral to integrate is often difficult. The microcontroller vendors often trade operating frequencies and system design flexibility against time-to-market requirements from their customers and overall lower system cost. Manufacturers have to balance the need to minimize the chip size against additional functionality.

Microcontroller architectures vary widely. Some designs include general-purpose microprocessor cores, with one or more ROM, RAM, or I/O functions integrated onto the package. Other designs are purpose built for control applications. A microcontroller instruction set usually has many instructions intended for bit-wise operations to make control programs more compact.For example, a general purpose processor might require several instructions to test a bit in a register and branch if the bit is set, where a microcontroller could have a single instruction to provide that commonly-required function.

Microcontroller typically do not have a math coprocessor, so fixed point or floating point arithmetic are performed by program code.

2.6 Programming environments:

Microcontrollers were originally programmed only in assembly language, but various high-level programming languages are now also in common use to target microcontrollers. These languages are either designed specially for the purpose, or versions of general purpose languages such as the C programming language. Compilers for general purpose languages will typically have some restrictions as well as enhancements to better support the unique characteristics of microcontrollers. Some microcontrollers have environments to aid developing certain types of applications. Microcontroller vendors often make tools freely available to make it easier to adopt their hardware.

Many microcontrollers are so quirky that they effectively require their own non-standard dialects of C, such as SDCC for the 8051, which prevent using standard tools (such as code libraries or static analysis tools) even for code unrelated to hardware features. Interpreters are often used to hide such low level quirks.

Interpreter firmware is also available for some microcontrollers. For example, BASIC on the early microcontrollers Intel 8052; BASIC and FORTH on the Zilog Z8 as well as some modern devices. Typically these interpreters support interactive programming.

Simulators are available for some microcontrollers, such as in Microchip's MPLAB environment. These allow a developer to analyze what the behavior of the microcontroller and their program should be if they were using the actual part. A simulator will show the internal processor state and also that of the outputs, as well as allowing input signals to be generated. While on the one hand most simulators will be limited from being unable to simulate much

other hardware in a system, they can exercise conditions that may otherwise be hard to reproduce at will in the physical implementation, and can be the quickest way to debug and analyze problems.

Recent microcontrollers are often integrated with on-chip debug circuitry that when accessed by an In-circuit emulator via JTAG, allow debugging of the firmware with a debugger.

2.7 Types of microcontrollers:

As of 2008 there are several dozen microcontroller architectures and vendors including:

- 68HC11
- 8051
- ARM processors (from many vendors) using ARM7 or Cortex-M3 cores are generally microcontrollers
- Atmel AVR (8-bit), AVR32 (32-bit), and AT91SAM
- Freescale ColdFire (32-bit) and S08 (8-bit)
- Hitachi H8, Hitachi SuperH
- MIPS (32-bit PIC32)
- NEC V850
- PIC (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24)
- PowerPC ISE
- PSoC (Programmable System-on-Chip)
- Rabbit 2000
- Texas Instruments MSP430 (16-bit), C2000 (32-bit), and Stellaris (32-bit)
- Toshiba TLCS-870
- Zilog eZ8, eZ80

and many others, some of which are used in very narrow range of applications or are more like applications processors than microcontrollers. The microcontroller market is extremely fragmented, with numerous vendors, technologies, and markets. Note that many vendors sell (or have sold) multiple architectures. In mid-2009, some consolidation is evident, with vendors pruning product lines.

2.8 Interrupt latency:

In contrast to general-purpose computers, microcontrollers used in embedded systems often seek to optimize interrupt latency over instruction throughput. Issues include both reducing the latency, and making it be more predictable (to support real-time control).

When an electronic device causes an interrupt, the intermediate results (registers) have to be saved before the software responsible for handling the interrupt can run. They must also be restored after that software is finished. If there are more registers, this saving and restoring process takes more time, increasing the latency. Ways to reduce such context/restore latency include having relatively few registers in their central processing units (undesirable because it slows down most non-interrupt processing substantially), or at least not having hardware save them all (hoping that the software doesn't then need to compensate by saving the rest "manually"). Another technique involves spending silicon gates on "shadow registers": one or

more duplicate registers used only by the interrupt software, perhaps supporting a dedicated stack.

Other factors affecting interrupt latency include:

- Cycles needed to complete current CPU activities. To minimize those costs, microcontrollers tend to have short pipelines (often three instructions or less), small write buffers, and ensure that longer instructions are continuable or restartable. RISC design principles ensure that most instructions take the same number of cycles, helping avoid the need for most such continuation/restart logic.
- The length of any critical section that needs to be interrupted. Entry to a critical section restricts concurrent data structure access. When a data structure must be accessed by an interrupt handler, the critical section must block that interrupt. Accordingly, interrupt latency is increased by however long that interrupt is blocked. When there are hard external constraints on system latency, developers often need tools to measure interrupt latencies and track down which critical sections cause slowdowns.

    - One common technique just blocks all interrupts for the duration of the critical section. This is easy to implement, but sometimes critical sections get uncomfortably long.
    - A more complex technique just blocks the interrupts that may trigger access to that data structure. This often based on interrupt priorities, which tend to not correspond well to the relevant system data structures. Accordingly, this technique is used mostly in very constrained environments.
    - Processors may have hardware support for some critical sections. Examples include supporting atomic access to bits or bytes within a word, or other atomic access primitives like the LDREX/STREX exclusive access primitives introduced in the ARMv6 architecture.
- Interrupt nesting. Some microcontrollers allow higher priority interrupts to interrupt lower priority ones. This allows software to manage latency by giving time-critical interrupts higher priority (and thus lower and more predictable latency) than less-critical ones.
- Trigger rate. When interrupts occur back-to-back, microcontrollers may avoid an extra context save/restore cycle by a form of tail call optimization.

Lower end microcontrollers tend to support fewer interrupt latency controls than higher end ones.

2.9 Comparison between PIC and Atmel 8051/8052:

Feature	**8051/52**	**PIC18XXX**
Program ROM(maximum space)	64k	2M
Data RAM(Maximum space)	256 byte	4K
Timers	3	4
I/O Pins	32	33
Serial port	1	1

Chapter # 3

SHORTEST PATH ALGORITHM

3 SHORTEST PATH ALGORITHM

(Dijkstra's algorithm)

3.1 INTRODUCTION:

Dijkstra's algorithm is called the single-source shortest path. It is also known as the single source shortest path problem. It computes length of the shortest path from the source to each of the remaining vertices in the graph.

The single source shortest path problem can be described as follows:

Let G= {V, E} be a directed weighted graph with V having the set of vertices. The special vertex s in V, where s is the source and let for any edge e in E, EdgeCost(e) be the length of edge e. All the weights in the graph should be non-negative.

Before going in depth about Dijkstra's algorithm let's talk in detail about directed-weighted graph.

Directed graph can be defined as an ordered pair G: = (V,E) with V is a set, whose elements are called vertices or nodes and E is a set of ordered pairs of vertices, called directed edges, arcs, or arrows. Directed graphs are also known as digraph.

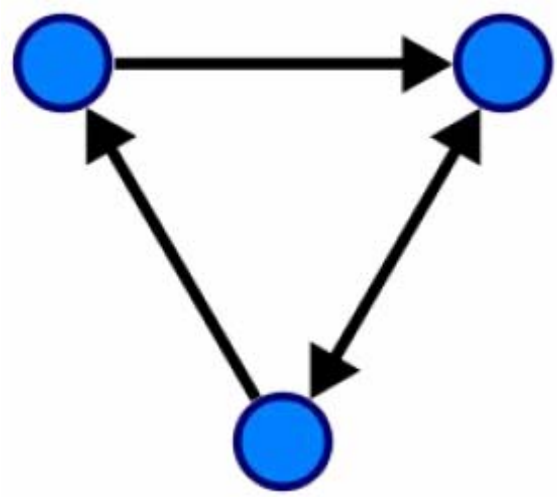

Figure1: Directed graph

Directed-weighted graph is a directed graph with weight attached to each of the edge of the graph.