March 2004

Mid-Semester Project Evaluation Report

Robotic Lower Limb for Above Knee Prosthesis

Project Guides:
R. K. Sharma (ECE)
Dr. R. P. Tewari (ICE)
Vicky Suri (ICE)

Students:
Sameer Singh 83 ECE 2k
Shuja Hussain 101 ECE 2k
Raman Agarwal 449 ICE 2k

Contents

Objectives

In a sentence, the objective of the project is to design and fabricate a robotic leg for humans which can be used for above knee prosthesis. It should confine itself to the following parameters.

- **Natural Looking Gait** – The leg should provide a natural looking human gait to the user, by using the inputs from different sensors. Thus the actuation has to be smooth and fast.
- **Customizable for Different Users** – The parameters of the gait, and the physical parameters of the leg, should be easily customizable for use for different individuals.
- **Automatic Operation** – The leg should operate using only the minimum interaction from the user. All tasks should be automated, and the required inputs should be taken from the sensors, and not from the user.
- **Low Weight** – The total weight of the leg should not be more than 3 kg for the comfort of the user.
- **Low Cost** – The project should not use unnecessarily expensive components, and priority should be given to cost.
- **Low Power Consumption** – Since the project is for mobile application, it shall include a portable battery. Thus the total consumption of the electrical components involved should be minimal.

Introduction

There are today about fifty different above knee prosthetic devices on the market. Many of these prosthetic involve:

- A socket for receiving and engaging the stump of the user.
- A frame extending down from the bracket and being pivotally connected to the bracket by a horizontal shaft. Bracket, shaft and frame together combining to form an artificial knee joint.
- A pylon and artificial foot connected to the base of the frame.
- And a means for controlling the knee joint by locking it to prevent it from buckling under load in the stance phase of a step, and freeing it in the swing phase of the step.

The knee mechanism of traditional above-knee prostheses can control the motion of the artificial leg only in the passive way, i.e. no feedback from the leg is provided to adjust the knee moment. With traditional design of the above knee prostheses, the amputee can only walk with a limited range of cadence. Further, a high percentage of mechanical work is done by the patient's lower extremity muscles due to compensation actions such as vaulting of the artificial leg to maintain gait stability.

For the development of new prostheses, bio-signal feedback to actively control the gait is necessary. Moreover, in order to automate the leg a microcontroller based system is designed which follows movements of the able-leg and enables the artificial-leg to move accordingly.

A revolutionary invention was the introduction of **Otto Bock's C-Leg**. The C-Leg was the world's first complete micro-processor-controlled prosthetic knee/shin system with hydraulic swing and stance phase control. In its design the electronic system monitors how the amputee is walking and creates smooth, harmonious movement of the prosthetic limb, similar to that of the sound leg, immediately adapting to different walking speeds and providing knee stability the moment it needed.

Another such invention (patented in United States in 1994) by Kelvin B James provides a system for controlling the rotation of a knee joint of an above knee prosthesis. The system employs a microprocessor, responsive to lower leg bending moment strain and knee angle measurements originating from sensors on the prosthesis to control a hydraulic damper through operation of a valve assembly, and thereby passively damp the rotation of the artificial knee joint in each of flexion and extension.

Other relevant patents in this regard are the French and the Russian patents which control above knee prosthesis by teaching sensors at the foot and the knee hinge respectively.

In India, research is going on to manufacture an artificial leg to provide the amputee with a natural looking gait in Defense Research and Development Organization, Ministry of Defense. But so far there is no commercially available smart leg which could emulate an able leg.

Design of the artificial leg, as proposed by us, presents a precise control of the knee movement. It utilizes a microcontroller which drives pneumatic cylinder and valves and provides control using feedback sensors. It is customizable for each user. This leg should offer various advantages over the conventionally used prosthetic leg in India, for instance the Jaipur Foot. This foot is non-automated, does not have any dampers, is cumbersome and has an unnatural gait. The robotic leg which we plan to make will have features to overcome these problems.

Our Approach

The approach used by us can be understood with the help of the block diagram shown below. The different main sections of the system have been marked out with grayed ovals. The various block of the project shall be described in brief below, and in detail further on.

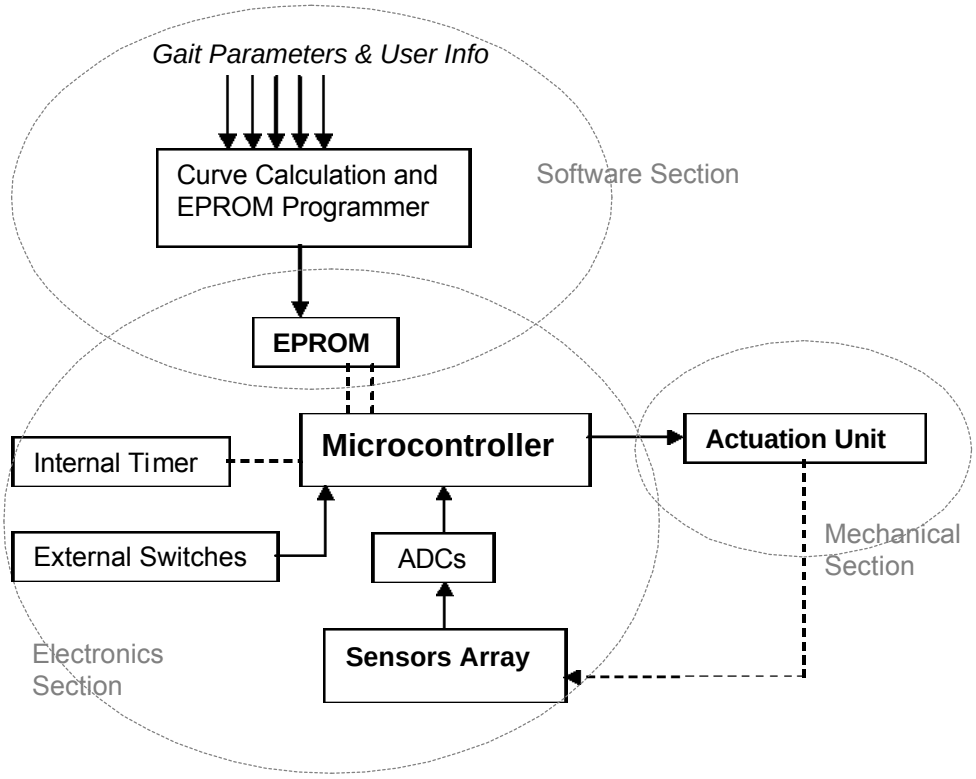


Fig: Main Control Block Diagram of the Project *showing the various components and the sections of the project.*

Software Section

This section primarily resides on a PC, in form of software. It includes primarily the curve calculation program to generate the curve from the user gait parameters, and a EPROM Programming program which writes this curve onto the EPROM in the microcontroller understandable form.

Electronics Section

The most important section of the project, it contains the microcontroller, the sensors and the associated data converters and interface circuits. Based on the gait curve available on the EPROM (internal to the microcontroller), the inputs from the sensors and other switches, the actuation unit is controlled to provide smooth and natural looking gait.

Mechanical Section

This section consists mainly of the actuation unit and the physical characteristics of the robotic leg. The actuation unit to be used is a pneumatic system which is fast as well as smooth. The physical characteristics include the size, the materials to be used, types of joints, placement of other components, etc.

In this way the project has been divided into three different sections. Work in these three sections is going in parallel. These sections shall be explained in detail in the following pages, and the work completed shall be mentioned. Various problems faced shall also be discussed.

Software

This section consists mainly of the three parts, namely the Gait Cycle, the Curve Calculation using Hermite Curves and the EPROM File. The Gait Cycle part consists of the introduction to the human gait cycle and basic terms and parameters of the human gait. The calculation of this curve according to the different parameters of the gait cycle using Hermite curves shall be described in the next part. Lastly, the EPROM Programming method will be explained along with the format of the EPROM user gait file.

Gait Cycle: Software

Walking relies on a repetition pattern of limb motion, under selective muscle control, to accomplish mainly four actions. Two of these actions, progression and weight-bearing stability, are essential. The supplementary functions, shock absorption and energy conservation, improve the quality of walking. Since our purpose is to develop a robotic leg for prosthesis, our priority will be the former functions.

The progression is achieved by synchronization of the gait cycle of both the legs, while the weight-bearing ability will automatically be incorporated once the right materials are used for the development of the robotic leg.

The gait cycle can be divided into two main phases, based on the force at the foot. The phase in which the foot is in contact with the ground is known as the stance phase. There is very little change in the angle of the knee. The other phase, when the foot is not in contact with the ground, is known as the swing phase, and has the maximum bending of the knee joint. The phases oppose each other during walking, i.e. when the right leg is in the swing phase, the left leg is in the stance phase, and vice versa.

The gait cycle (in reference to the knee angle) differs from person to person in respect to the following main parameters:

- Stride Length – Stride Length is the total length between two consecutive contacts of one leg with the ground.
- Velocity – This is the speed of progression of the person. It varies greatly from person to person and is one of the most important parameters. It is used to calculate the stride frequency, and hence the time for one stride.
- Peak Knee Swing Phase Flexion – The maximum angle achieved by the person during the swing phase. It is usually $70^o \pm 10^o$.
- Peak Knee Stance Phase Flexion – The maximum angle achieved during the stance phase. It is usually $23^o \pm 5^o$.
- Peak Knee Extension in Late Stance – The minimum angle achieved during the stance phase, i.e. when the stances are changing. It is usually $2^o \pm 4^o$.

These parameters have been marked out in the figure.

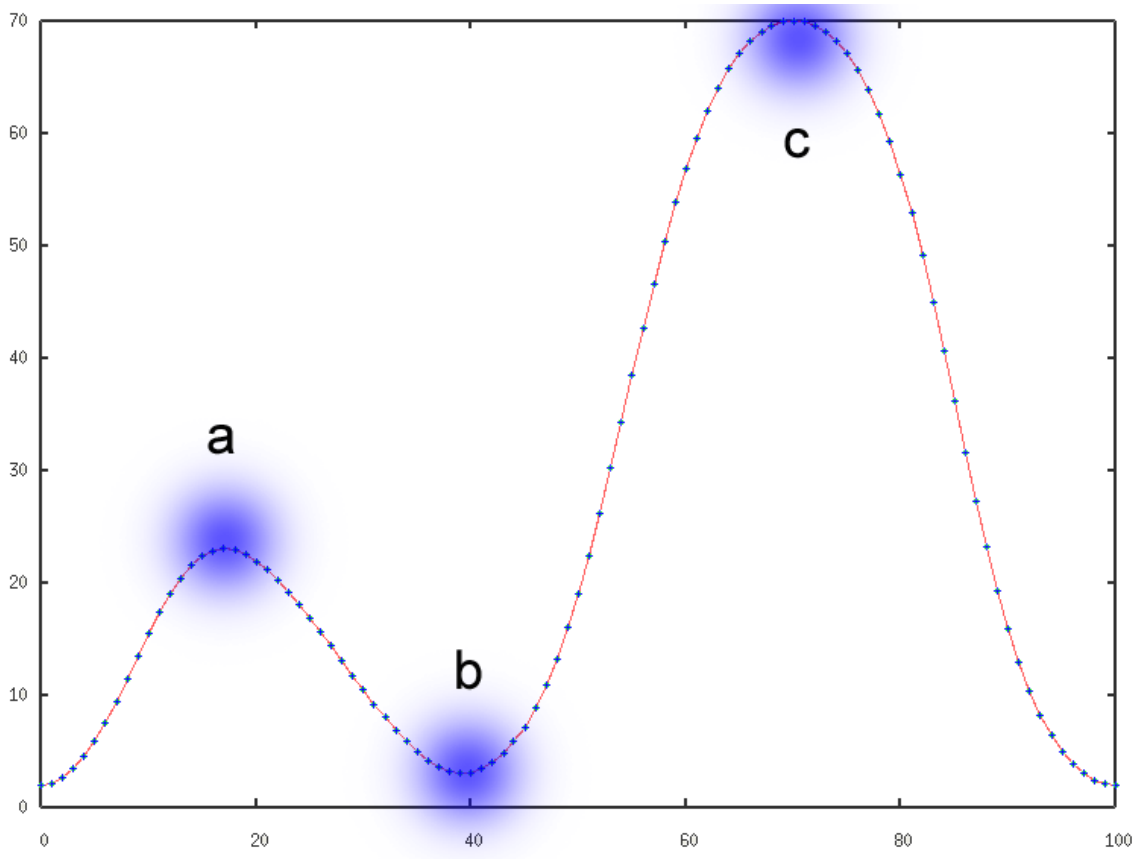


Fig: Gait Cycle Parameters for the curve
a) Peak Stance Phase Flexion b) Peak Late Stance Extension
c) Peak Swing Phase Flexion

Curve Generation: Software

As was described in the last part, each user has a number of parameters defining his gait. Some parameters are used only for the purpose of the mechanical design of the leg, like stride length, and we shall not be discussing those in this part. Also time related parameters like the total stride time shall also not be used since they are only for the purpose of the expanding or contracting the curve, and they shall be used by the microcontroller for its operation. Instead the curve shall be stored in terms of the percentage of the gait cycle.

Thus the problem now remains to calculate the curve shape according to parameters like, peak swing phase flexion angle, peak stance phase angle, peak extension in late stance angle, etc. These points have been marked out on the graph shown below. Along with these, for the correct calculation of the curve, we would require time (in %) of when these angles occur, and another parameter (peak extension in early stance angle). If these values cannot be calculated from the patient, then the average values of these parameters can be taken.

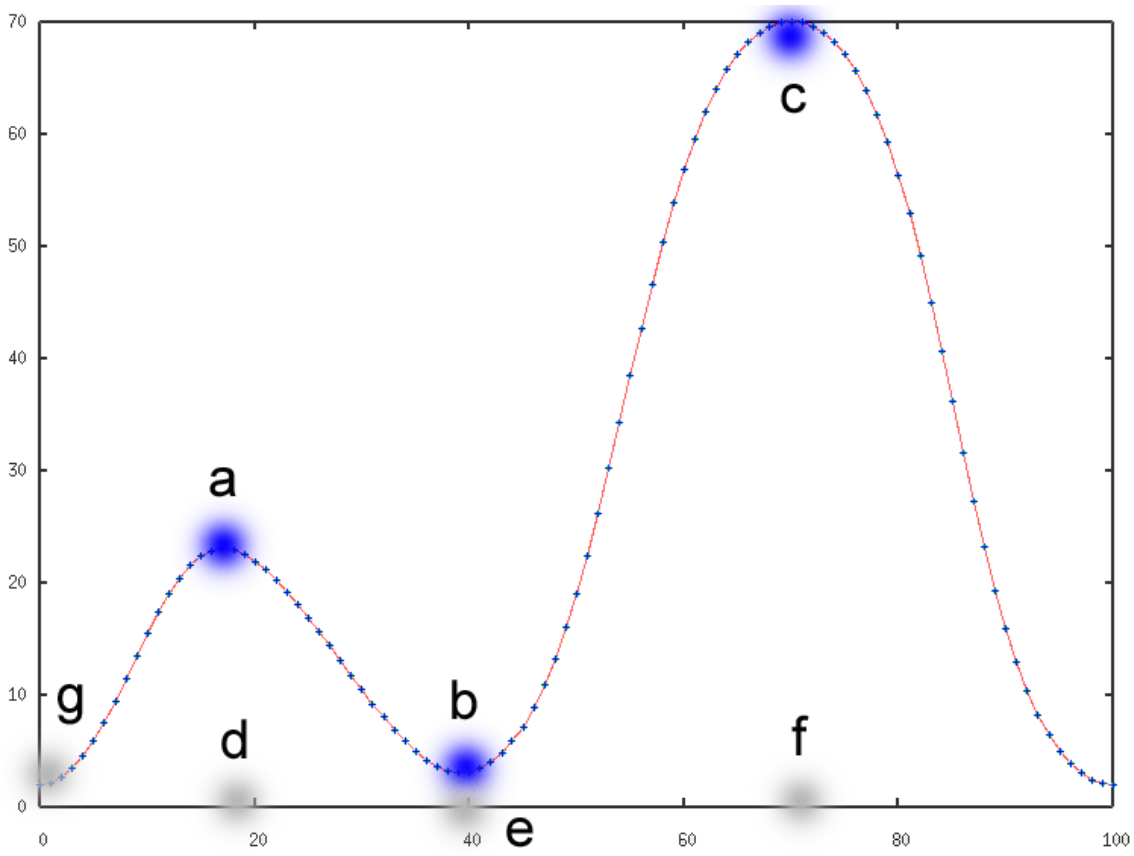


Fig: Gait Parameters Along with the Curve Parameters
a), b) & c) are the parameters as explained in the last part; d), e) & f) are their respective time; g) Peak Extension in Early Stance Phase

Once these parameters have been obtained, the task now is to calculate the curve. This was done by using the curve generating algorithms used in graphics. Graphics has a wide application of these curve plotting algorithms, like use in path of cameras or objects, smoothing of intensities or colors, etc. Thus a lot of research has gone into this field for generating the most efficient algorithm. We shall be borrowing these algorithms for our application.

The curve generating algorithms are mostly parametric cubic curves, i.e. they consist of cubic equation between two points (*piecewise polynomial*). The curve is not directly a function of *x* or *y*, but a function of a parameter *t.* The *x* and *y* coordinates are cubic polynomial functions of *t* between the adjacent points. The cubic polynomial between two points should also be differentiable to the rest of the curve.

The choice of cubic equations was inevitable. Higher degree curves introduce unnecessary wiggles and require more computation. On the other hand, lower degree curves give too little flexibility in controlling the shape of the curve, i.e. you cannot specify the derivatives and the point through which it should pass.

Hermite Curves

Hermite Curves are one type of these curves which require eight inputs for a 2D curve generation, namely, two interpolation points (two coordinates each) and the derivatives at these points (direction and magnitude). If the points are termed as P_{1x}, P_{1y}, P_{2x} & P_{2y} and the derivatives are termed R_{1x}, R_{1y}, R_{2x} & R_{2y}, then the curve between these points will be given by (where t varies from 0 to 1):

$X(t) = (2t^3 – 3t^2 + 1)P_{1x} + (-2t^3 + 3t^2)P_{2x} + (t^3 – 2t^2 + t)R_{1x} + (t^3 – t^2)R_{2x}$

$Y(t) = (2t^3 – 3t^2 + 1)P_{1y} + (-2t^3 + 3t^2)P_{2y} + (t^3 – 2t^2 + t)R_{1y} + (t^3 – t^2)R_{2y}$

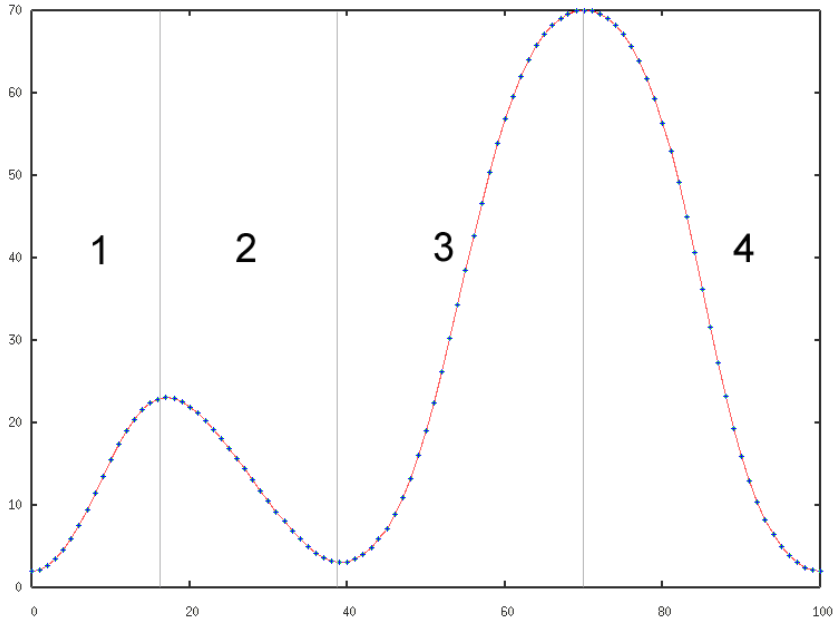


Fig: Gait cycle divided into four Hermite curves

The gait cycle has been divided into four curves like this as shown in the figure, with the derivatives pre-calculated. For a smooth curve the derivative of the end point of the previous curve should be same as that of the starting point of the curve. Extra points were added before and after the curve to ensure smooth cycling of the gait cycle. This extra curve is later clipped.

EPROM File: Software

Once the curve has been calculated, it has to be converted to the microcontroller usable format. This is achieved by using the following steps:

1. The curve generated has to be clipped since extra points had been added for smooth curve cycling.
2. The curve has to be sampled for a final generation of a hundred points (corresponding to the percentages). To ensure proper sampling, it should be ensured that the precision of the increment of t is high enough.
3. The value of the angle at each point has to be converted from the floating point to one byte. The total range of angles (0.0 to 90.0) is scaled and rounded off to one which can fit into a byte (0x00 to 0xFF).
4. Additional information is added to the curve like the user initials, mean stride time, a date stamp, user weight, weight threshold etc. creating a final EPROM file of maximum 128 bytes.
5. Finally the EPROM file has to be programmed into the microcontroller, the method of which shall be explained in the Electronics section.

Electronics

This section describes the circuits and the microcontroller in detail. It includes descriptions of the sensors and their use, the associated circuitry, the microcontroller connections, the microcontroller code flow chart and EPROM programming method.

Sensors & Switches: Electronics

Knee Angle Sensor

This sensor shall return the value of the angle of the knee. The sensor to be used is a resistive type angle sensor. It gives out an analog signal proportional to the current angle. The range of the sensor should be at least 0-90, which is not a problem since mostly they have a range of 0-180 or 0-360. This angle input is used to move the actuation in the desired direction.

Force Sensor

An input has to be taken in by the microcontroller which should correspond to the force on the leg. This value helps signal the start and the stop of different phases of the gait. The range should be approximately 0-100kgs. For this purpose various sensors were studied, namely piezoelectric sensors, piezo-resistive sensors (strain gauges), pressure sensors, etc. None of them were found suitable enough for the purpose due to their weight, dimensions or force ranges. The only one which came close to the requirements were the strain gauges, which shall be used.

Vibration Sensor

A vibration sensor shall be attached to the socket which holds the amputation stump. This sensor shall sense movements in the muscle which correspond to the initializing or stopping of the walking movement. Multiple vibration sensors can be used to detect more movements like running, sitting, and climbing. These phases can then be incorporated later.

Switches

Switches shall be placed to give more inputs to the microcontroller. A switch shall be used to differentiate between the programming and the program execution mode. A potentiometer type knob shall be used to specify the relative speed of the program execution. An emergency actuation release key has to be pressed for emergency situations.

Analog to Digital Converters

The ADCs used have to be mixture of the internal and the external ADCs (relative to the microcontroller). The internal ADC of the microcontroller is slow in response, so can be used for switches and sensors which do not require high speed, like the speed knob, vibration sensor and the switches. External ADCs shall be used for high speed applications like for the force sensor and the angle sensor.

Microcontroller: Electronics

One microcontroller is to be used in the project. The requirements from the microcontroller were as follows:

1. EPROM – At least 128 bytes
2. FLASH Program Memory – Enough for a medium length program
3. In built ADCs – At least 3; with maximum conversion time 250ms
4. I/O Pins – At least 25
5. Inbuilt Timer – Preferably 16 bit
6. In System Programming
7. Low cost & Availability

Based on these requirements, the microcontroller decided is the AT90LS8535, from the AVR range by Atmel®. Its characteristics are as given below.

- Flash – 8 KB
- EPROM – 512b
- I/O Pins – 32
- Timers – 2 8-bit; 1 16-bit
- ADCs – 8 10-bit
- Analog Comparator

The ports shall be divided as follows:

- Force Sensor – 8 pins
- Angle Sensor – 8 pins
- Speed Switch – 1 pin (ADC)
- Emergency stop switch – 1 pin
- P/E (Programming EPROM / Execution) Switch – 1 pin
- Vibration Sensors – 3-4 pins
- Status LEDs – 3 pins
- Actuation Unit – 4 pins
- Programming Mode – 8+3 pins (multiplexed with other pins)

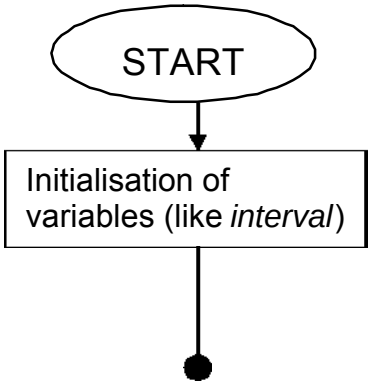
Code Flow Chart

Before the chart is given, the different components of the program shall be explained. It shall consist of a number of different modules running in synchronization of each other. For this purpose an interrupt-like environment will be developed, which, according to the status of some of the registers, shall direct the program flow to specified areas. Interrupts within the microcontroller shall be used wherever possible, but is not possible, then code emulating interrupt behavior (like ISR address, return address, etc.) shall have to be written.

The flow chart has been drawn assuming all interrupts have to be emulated, and no interrupts exist in the system. The kinds of routines and interrupts that shall be generated are as follows:

- **Timer:** Generated every (stride time)/100 of a second by reading the timer value, which has been named *interval*. This interrupt updates the angle to be achieved by the actuation from the EPROM.
- **Force Threshold:** Whenever the input force crosses the threshold, this interrupt shall be generated to decide the phase of the gait. If this interrupt is not generated, the program will stop executing till it is. For example, if the user stops walking, then the program shall not turn the leg till the user lifts it again.
- **Vibration Input:** This input decides the starting or stopping of the microcontroller gait routine. It can be used as a toggle, or inputs from different sensors can be used.
- **Emergency Release:** The actuation, in this routine, has to be released, i.e. all the valves have to be opened, leaving the cylinder free to move.
- **Update Interval:** This routine shall update the value of the interval of the timer if the speed switch has been rotated to change the value.

These routines, and the program flow, have been shown in the flow chart. Note that in this section we are only discussing the Execution Mode, and not the Programming Mode.



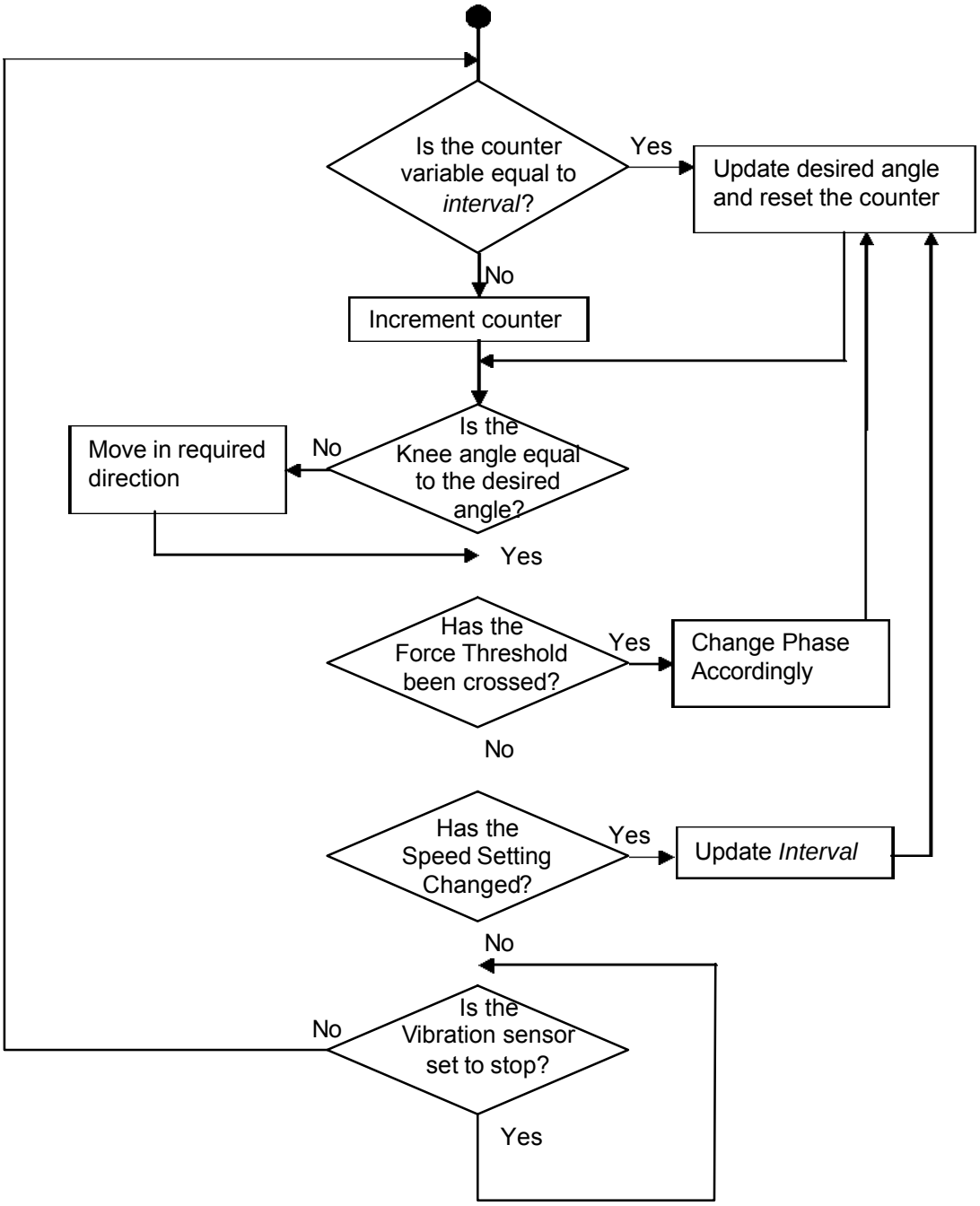


Fig: Flow Chart for the Microcontroller showing various different routines and interrupts

EPROM Programming: Electronics

The EPROM Programming comes into picture when the P/E switch is set to P. The microcontroller assumes that it is connected to the PC via the 8+3 pins of the parallel port. The 8 pins of these act as data inputs (*D0-D7*) to the microcontroller, while out of the rest three, 2 act as inputs (Programming On (*Pon*) and toggling clock (*Clk*)) and one as output (Toggling Acknowledgement (*Ack*)). The number of bytes to be transferred is fixed, and thus the microcontroller stops programming as soon as it writes the required number of bytes. There is no addressing involved since the whole EPROM file has to be written from the starting address.

The microcontroller starts reading the data only when the *Pon* pin is high. It recognizes a *new* byte when the *Clk* has toggled from the previous value. After every write onto the EPROM, the microcontroller toggles the *Ack* pin. At every write, the microcontroller increments an internal address register. The PC, i.e. the programmer, sends a new byte and toggles the *Clk* whenever the *Ack* is toggled. If the software needs to resend the whole file, all it needs to do is set the *Pon* to low for a while and then set it to high again to reset the programming mode of the microcontroller. Thus there is efficient transfer of data.

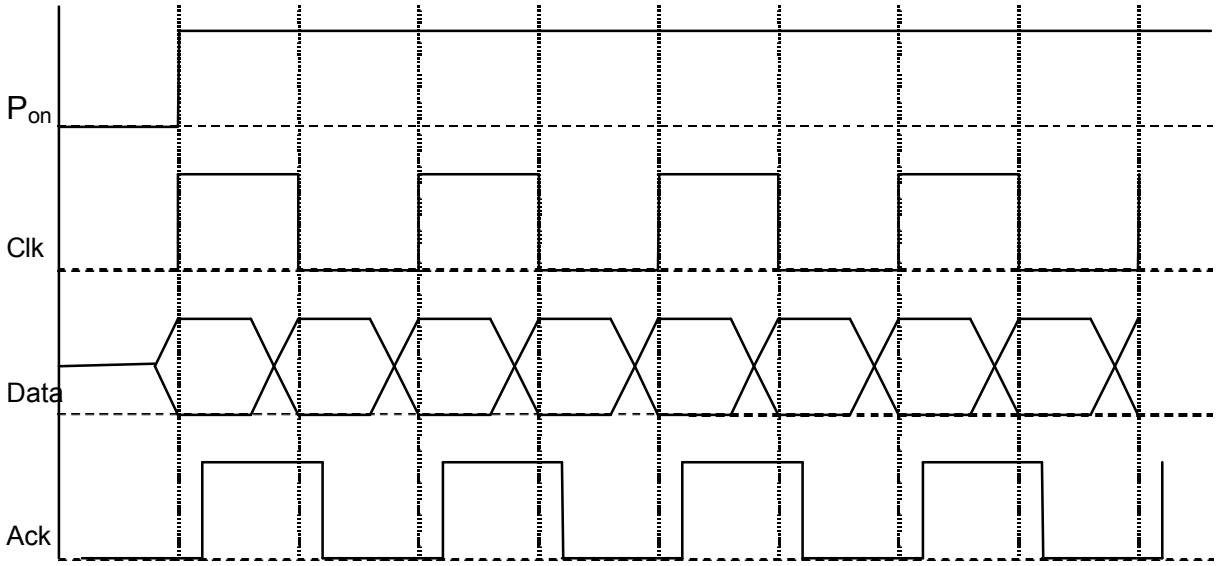


Fig: Timing Diagram for the EPROM Programmer showing the various signals

Mechanical

In our mechanical design of the leg, knee is controlled with a pneumatic piston which in turn is operated by electro-pneumatic valves (double acting solenoid valves). This assembly was adopted keeping in mind, the requirement of precise movements of the knee-joint. It is elaborated in the figures below.

Two metal plates or bars are hinged together, hinge being the knee joint. A pneumatic cylinder is also hinged on both these plates using normal screws such that the piston can rotate about them when the person is walking. When the artificial leg is absolutely straight then the piston is fully open i.e. it is in its full stroke length. Working of this assembly is analogous to working of a scissor. When the piston extends, the knee also extends, and when it is pushed in the knee flexes or bends.

The to and fro movement of the piston is governed by double acting solenoid valves. In our mechanism two 3/2 solenoid valves are used, one on either side of the cylinder. Initially both the valves are connected to the compressor so that the piston is in the "locked state" (since it is experiencing pressure 'P' from both the sides). Now, if it has to move towards any one side then valve of that particular side is connected to exhaust. Duration for which air in the chamber is exhausted depends on how much the piston has to translate or rather how much the knee has to bend. Opening and closing of the valves is governed by digital pulses from the microcontroller. It should be noted here that since we are using a pneumatic mechanism air needs to be exhausted through flow control valves, so that it does not gush out quickly.

Earlier attempts

While designing the scheme for movement of the knee joint, we had earlier considered using hydraulic piston and valves since they could provide better precision. But a major drawback faced using hydraulic mechanism was the weight of the entire equipment. The hydraulic compressors, commercially available, are quite cumbersome besides, the fluid which will also add its own weight on the leg. Therefore incorporating hydraulic control scheme would make the leg heavy, leave alone the weight of the person that it has to bear. On the other hand, in the pneumatic assembly air has negligible weight and pneumatic compressors are also light, thus making it suitable for our requirement.

Additionally, while simulating the above described mechanism (using hydraulic scheme) in the laboratory we had used a 4/3 double acting solenoid valve. (Working of the valves is described in the next section). In the simulation, a C++ program was written which controlled the movement of the hydraulic piston. The data (bit stream) was sent through the parallel port of the PC which switched ON/OFF the corresponding relays which in-turn actuated the necessary valves.

Valves: Mechanical

1. **Single Acting 3/2 Valves**

By 3/2 valve we mean a valve having 3 ports (a pressure port, an exhaust port and an output port) and 2 positions (P & E). Working of this valve is simplest of all and it is quite cheap in cost too.

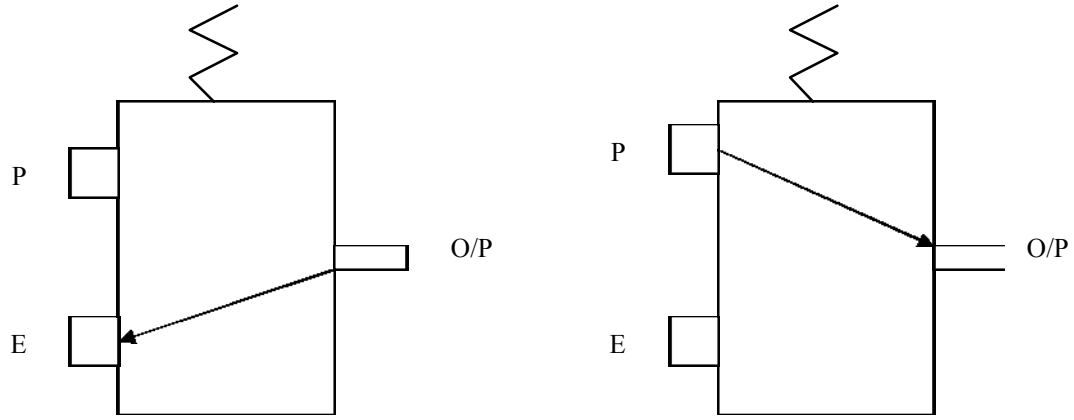
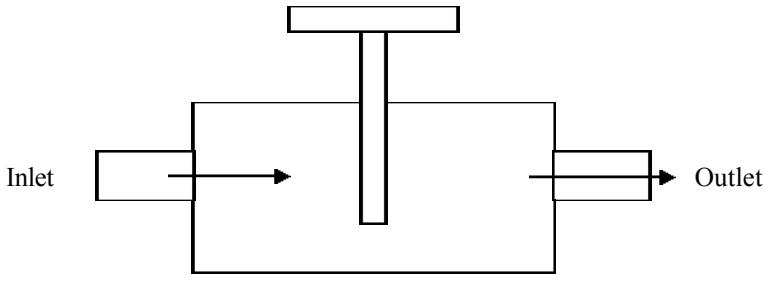


Fig: Working of a single acting 3/2 pneumatic valve

Commercially this valve is available as an NO or an NC configuration. An NO configuration means that when valve is not actuated the output is connected to the exhaust and as soon as the valve is actuated output port gets connected to the compressor. It should be noted here that the process depicted in the figure corresponds to a spring returned solenoid valve. If the valve is not spring returned the output port's connection depends upon which solenoid is actuated.

2. Flow control valves



Working of these valves is straightforward. Depending on the amount of screw that is into the cylinder, the air goes from one inlet to another. This valve is required for precise movement of the piston since it monitors the flow of air.

Working of the Leg

The figure below shows the profile view of the artificial leg based on our design. Its working is analogous to the working of a scissor. When the piston moves upwards, it pulls the two limbs of the leg towards each other, thus providing the necessary rotation of the knee joint. The cylinder and the piston have to be hinged on the leg as shown in the figure. This is necessary because when the limbs will move towards each other there will be lateral movement of the cylinder too. Thus, in order fix the cylinder at the same position (i.e. fixed to the leg) it has to be hinged at both ends.

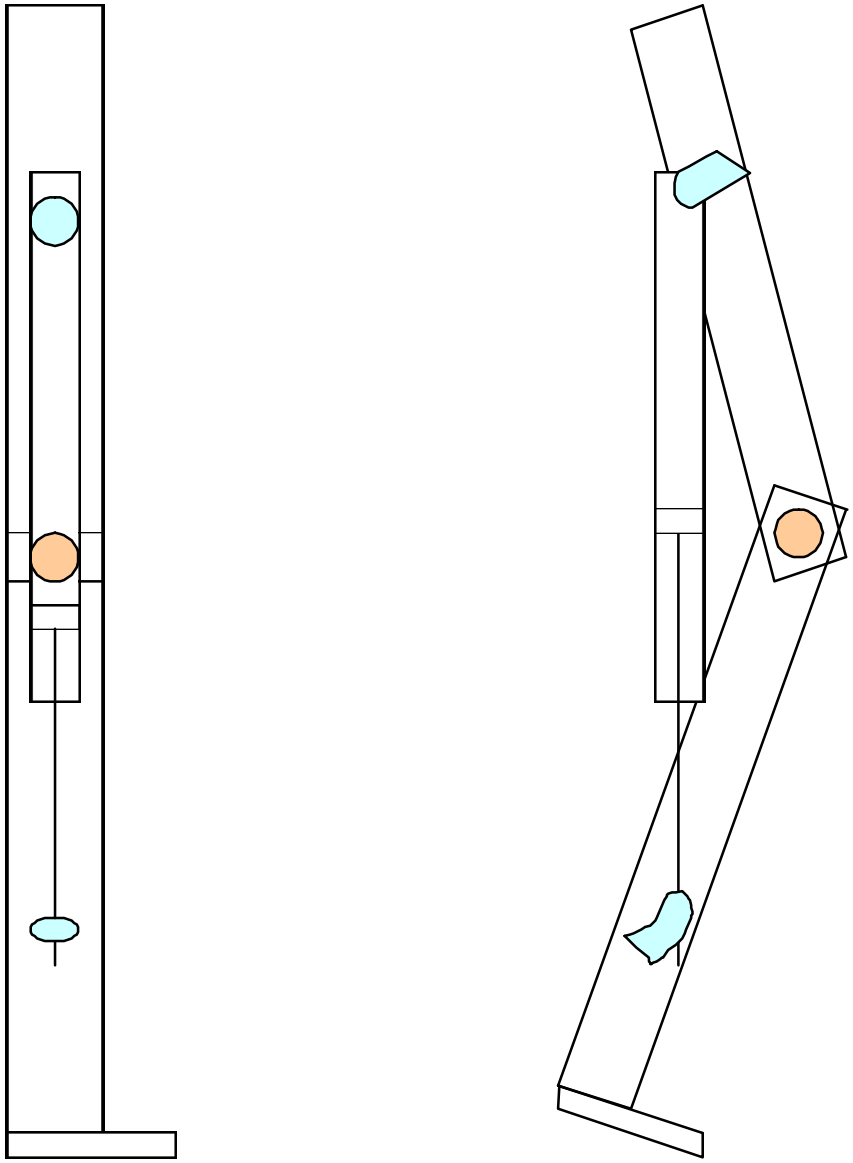


Fig: Movement of the leg with respect to the cylinder.

Interfacing the Valves

During our simulation we used the circuit as shown below to interface PC's parallel port with the hydraulic/pneumatic system.

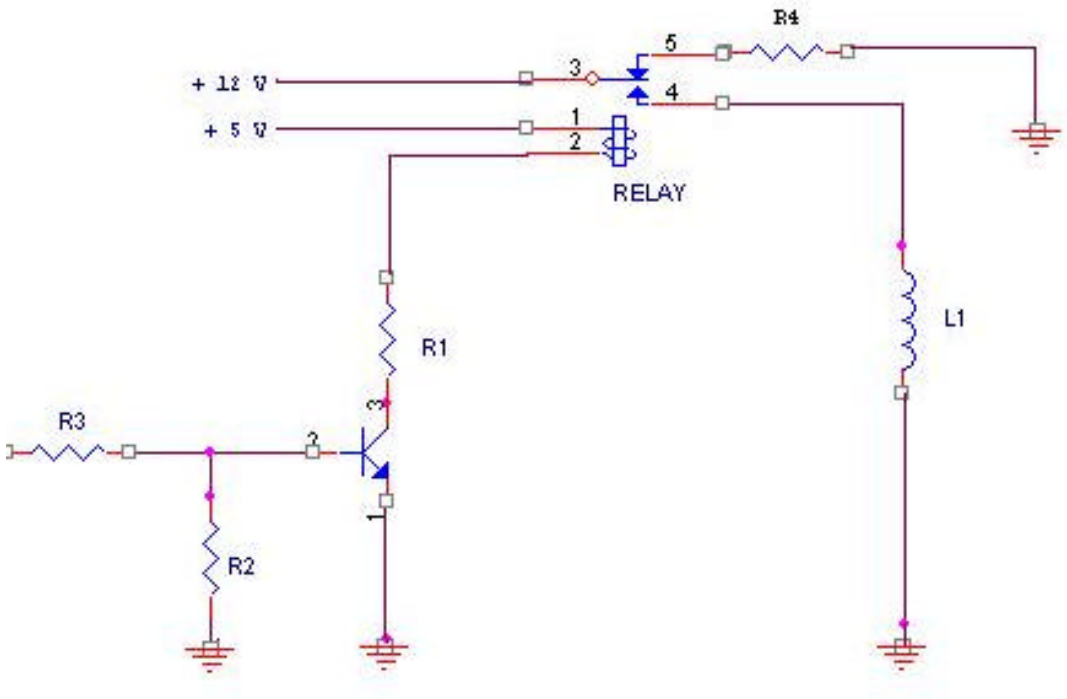


Fig: Interfacing Circuit between PC's parallel port and the pneumatic system during simulation exercise.

The transistor works as a switch here. A high bit from the parallel port switches it 'ON' thereby switching 'ON' a relay. This relay in turn actuates the corresponding valve to move the piston in the desired direction. The biasing of the transistor is required to ensure it works in the linear range. The transistor selected had â=100 so that current can be amplified up to 100 times. This particular â was preferred because the parallel port can source only 5mA of current and the relays used required 0.5A of current for driving the valves.

It should be noted that the bits from the PC can be sent with appropriate delay to ensure slow and precise movement of the piston.

Conclusion – Progress

In the first phase of the project our work had been developing a basic design of the artificial leg in terms of both mechanical and electrical aspect. This involved initial simulations of hydraulic system and later a pneumatic system using simple C++ code. Algorithms were written which could generate gait cycle of an amputee using seven parameters available from a database, and then store it in a file format suitable for the microcontroller.

Simultaneously, ground work on the components to buy and their vendors was carried out. Sensors are being purchased from a UK based worldwide distributors of electronics, electrical and industrial components. Quotations of the required angle sensors had been mailed to the relevant. As soon as the sensors are available and the microcontroller bought interfacing circuits shall be made. Additionally, manufacturing of the leg will continue side by side.

The table below describes the project progress as of 2nd week of March, 2004.

Project Section / Part	Progress Description	Progress (in %)
Software		
- Curve Calculation	Complete, Except for the GUI	**90.0**
- EPROM File	Complete, Except for the GUI	**90.0**
Electronics		
- Sensors	Selection and Ordering Done	**10.0**
- Microcontroller Circuit	Selection, Design and Ordering Done	**10.0**
- Microcontroller Code	Design done and Started	**33.3**
- EPROM Programming	Design done	**10.0**
Mechanical		
- Pneumatic System	Complete	**100**
- Interfacing with PC	Complete	**100**
- Interfacing with μC	Design done	**10.0**
- Mechanical Leg	Design done	**10.0**

References:

- Clinical Gait Analysis – *Davis, R.B.*
- The Mechanics of Gait – *Perry, J.*
- Gait Analysis – *Kaufman, K.R.*
- Fuzzy Control of Electrohydraulic Above-Knee Prostheses – *Ju, M.S., Yi, S.H., Tsuie, Y.G. & Chou, Y.L.*
- System for Controlling Artificial Knee Joint Action in an Above-Knee Prosthesis – *James, K.B.* (US Patent No.- 5571205)
- Otto Bock website – *www.ottobockus.com*
- Computer Graphics: Principles & Practice – *Foley, J.D., van Dam, A., Feiner, S.K. & Hughes, J.F.* (Pearson Education)
- Farnell InOne Catalogue – *www.farnellinone.com*
- A Course in Mechanical Measurements and Instrumentation – *Sawhney, A.K. & Sawhney, P.* (Dhanpat Rai & Co.)
- Programming and Customizing the AVR RISC Microcontrollers – *Gadre, D.V.* (McGraw Hill)
- Festo Pneumatic Systems website – *www.festo.com*