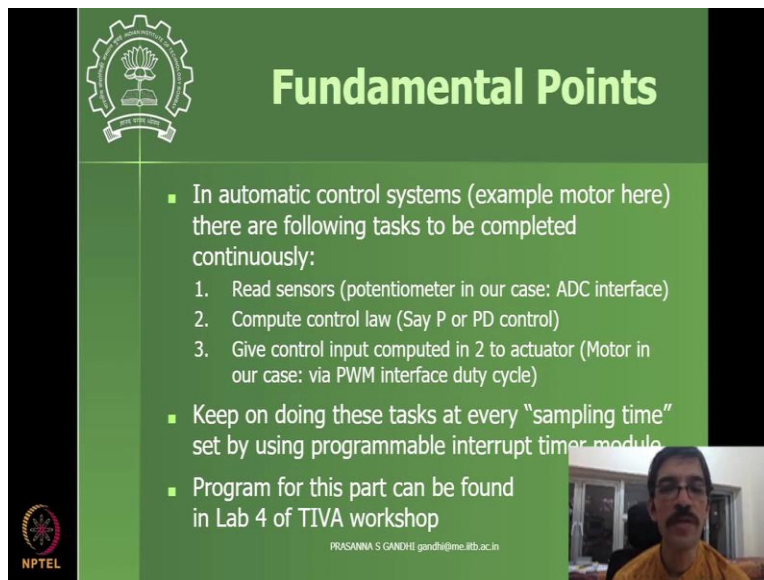


**Design of Mechatronic Systems**  
**Professor Prasanna S. Gandhi**  
**Department of Mechanical Engineering**  
**Indian Institute of Technology, Bombay**  
**Lecture 34**

**Closed Loop Control Implementation in Microcontroller**

So, now we will see in the second part of the lecture today the fundamentals for implementation of closed loop control, this we will need for this practical kind of part in the assignment and we will talk little about more it is very very important topic for practical implementation with your microcontroller. So, please go through this lecture meticulously to understand like lot of this fundamental conceptual details, this is more out of a philosophical level and through that you will be able to kind of use these ideas and things that are talked about for actual making sure your motor runs in the closed loop the way we decided to happen. So, let us get into more details here.

(Refer Slide Time: 01:13)



The slide features a green background with a white gear icon containing the IIT Bombay logo in the top left. The title 'Fundamental Points' is in large white font. Below it, a list of tasks is presented in white text. A small video inset in the bottom right shows Professor Prasanna S. Gandhi. The NPTEL logo is in the bottom left, and the professor's email address is at the bottom center.

- In automatic control systems (example motor here) there are following tasks to be completed continuously:
  1. Read sensors (potentiometer in our case: ADC interface)
  2. Compute control law (Say P or PD control)
  3. Give control input computed in 2 to actuator (Motor in our case: via PWM interface duty cycle)
- Keep on doing these tasks at every "sampling time" set by using programmable interrupt timer module
- Program for this part can be found in Lab 4 of TIVA workshop

PRASANNA S GANDHI gandhi@me.iitb.ac.in

So, this fundamental principle here for any automatic control system which we are talking with respect to the example of motor for our case, there are these following three tasks that happen continuously, one is reading of the sensors, so it is not potentiometer it is an encoder in our case, so this, please update that this is encoder in our case not a potentiometer. So, we read a sensor which is encoder in our case, then compute control now which is maybe P, PD kind of a control.

And that we have seen in the feedback concept of feedback and or maybe you might have seen some of these control algorithms in your automatic control kind of courses previously and then we need to give this control input to the actuator. So, how do you give these? We give this via PWM interface. Specifically we keep on changing the duty cycle depending upon what is completed control input and every step of the sampling.

So, this is running continuously every periodic kind of time. So, say, so that is called like sampling time and how do you set up the sampling time which we will see that okay that sampling time is set up by using a interrupt timer module. So, typically all microcontrollers will have this interrupt timer, programmable interrupt timer module which will allow you to set up this interrupt service routine.

So, which will run every sampling time that you have set in that routine. Now, let me explain this concept what is this interrupt and timer together would work, so concept of interrupts and presuming you know just to recap I give you the that okay you are reading a book and your mom calls, and your phone is ringing what you hear the phone and I hope you want to pick up that call, and what you do is you put a bookmark in the book and go pick up the phone, of course I am mentioning that you want to take up the call and then you reply to the call, you finish the call up and then again come back to the same bookmark and like open the book and start reading it.

Exactly the same kind of concept is there for the interrupts. So, microcontroller is kind of doing some task and on some pin this some signal comes, some pluses comes and that was they say this like interrupt pin specifically designed for that purpose, so on a interrupt pin when this pulse comes microcontroller will halt the current task for a moment and it will serve some other tasks which is defined for that particular pin.

So, that task will be called as interrupt service routine. So, that task will be executed and then again microcontroller will go back to its original point or your main program and starting repeating from where it has halted previously. Now, these pulses that are coming on this pin if every time this pulse comes the microcontroller has to execute some task which is called interrupt service routine.

And whatever these three operations I have mentioned here, these are called three operations they will be typically written in the interrupt service routine. So, they will be executed every

sampling time, now for them to get executed every sampling time this pulse needs to come on the pin at every sampling instances.

So, typically these policies are generated on some kind of a timer module we will do that, so counter timer module will have a job to develop this kind of a pulse every whatever sampling time instance and this pulse is internally connected you do not need to do any connections nothing in your microcontroller, this modern microcontrollers, you do not need to do nothing, it is software only the software that you need to kind of write because these pins are already internally connected.

So, that these pulse will get generated in the timer module and it will go on the interrupt pin and this purchase now are programmed to come regularly in some kind of the way you have a square waveform which is periodic these pulses are coming now in a periodic kind of a way. And when these pulses are coming every pulse comes up with these three tasks, or a every pulse comes you execute the your interrupt service routine will get executed and you keep on doing this task.

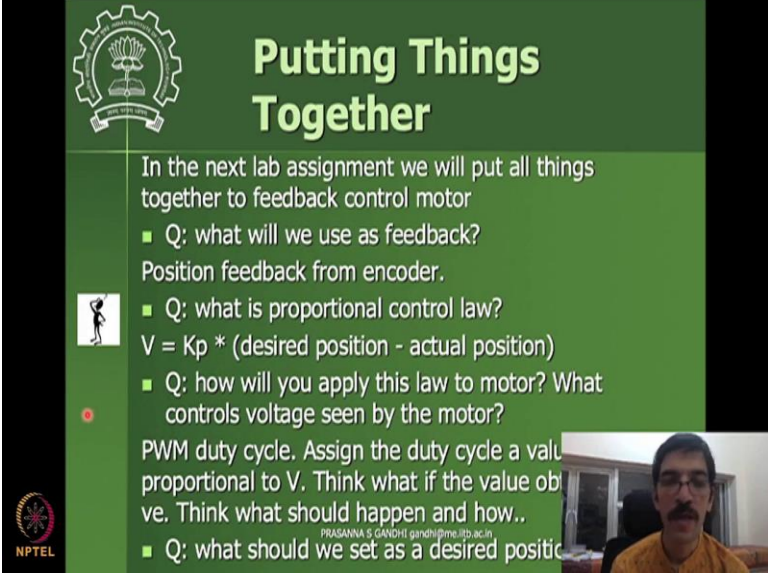
And how do you kind of program for this part of setting up sampling time or programmable interrupt timer module to program to give you some kind of a facility to have something written in the interrupt service routine and getting executed every sampling time, so you can do that first so that you can set up your sampling time properly, and then now you can slowly start putting things in the interrupt service routine of this module.

So, for example if you have written PWM program you need to kind of see what are the parts of the pyramid program which can go in the main program and what are the parts we should come in the interrupt service routine you can bifurcate for example you just want PWM duty to be changed here, so all other commands in the PWM routine should not like you know the other register they should not change or other functionalities should not change.

So, only the functionality which is giving you the handle over PWM duty cycle that can come in the interrupt service routine, like that in you think what will be coming from the encoder interface into your first reading sensor kind of part, like that you need to start thinking and programming. And you do not need to do any for loop this is very important, do not set up any for loop for doing this once you are putting these things in the interrupt service routine you are rest assured that they will run every sampling instance.

They will run continuously over and over again you do not need to put any for loop whatsoever for this purpose. We will discuss like in more detail, why we keep this as a interrupt based sampling time programming as we discuss the sampling and other issues in more detail. So, we want every fixed known sampling time for the operation to happen if the periodically and this our job will be done.

(Refer Slide Time: 08:47)



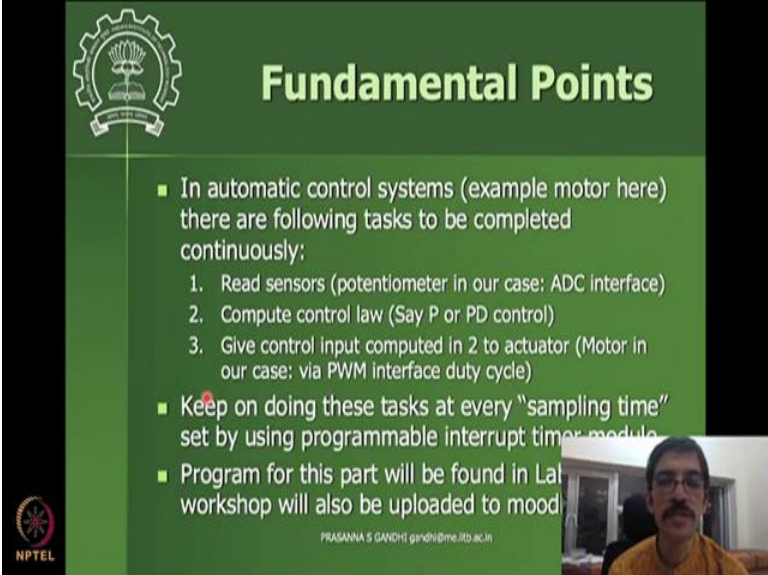
## Putting Things Together

In the next lab assignment we will put all things together to feedback control motor

- Q: what will we use as feedback?  
Position feedback from encoder.
- Q: what is proportional control law?  
 $V = K_p * (\text{desired position} - \text{actual position})$
- Q: how will you apply this law to motor? What controls voltage seen by the motor?  
PWM duty cycle. Assign the duty cycle a value proportional to V. Think what if the value of V is positive. Think what should happen and how..
- Q: what should we set as a desired position?

NPTEL

PRASANNA S GANDHI gandhi@me.iitb.ac.in



## Fundamental Points

- In automatic control systems (example motor here) there are following tasks to be completed continuously:
  1. Read sensors (potentiometer in our case: ADC interface)
  2. Compute control law (Say P or PD control)
  3. Give control input computed in 2 to actuator (Motor in our case: via PWM interface duty cycle)
- Keep on doing these tasks at every "sampling time" set by using programmable interrupt timer module
- Program for this part will be found in Lab workshop will also be uploaded to moodle

NPTEL

PRASANNA S GANDHI gandhi@me.iitb.ac.in

Now, so to put all these things together here we are using the encoder position feedback coming from the encoder. So, this is the first part here, reading sensors in encoding in this case and then

like will we use some kind of proportional or proportional derivative control now, so this expression typically will be have this kind of a form.

So, you will have some kind of a desired position and you have actual position and these things will be this desired position as you see later we will see what kind of a form we should have for the desired position, so that we keep on observing once we have implemented we keep on observing or control is how our controller is performing those kind of things we will see.

Then how do we apply this to controller to the motor? So, see this a control expression may give some negative value, depending upon what are the values of these, so how we when if the value is coming negative, how do you see PWM duty is only positive values 0 to 100, then we need to kind of do something to control motor, so we need to kind of give something to kind of reverse the direction of the motor, if the direction here the computation is negative that is what we need to do. So, you know how to do the direction reversal.

So, those are the things that we need to implement. So, that is what we will discussing what if this value is negative we need to use the NA in V pins of the driver to get reversed to change the direction of the motion. Then again this question is what should be set as desired position for the for this job to happen properly.

So, because if we just put some kind of a constant value in desired position you think what will happen and then we want to observe this happening properly then we need to kind of set these values in the appropriate way. So, these are all the questions that you ponder over and think yourself and see what solutions come to your mind for this and then you proceed with the next part. So, this pondering over for you is important here.

(Refer Slide Time: 11:20)

**Putting Things Together:  
proportional control**

$V = K_p * (\text{desired position} - \text{actual position})$

- Q: what will happen if we keep the desired position constant?  
motor will move from whatever current position to desired and we will not have enough opportunity to observe how control is working. We will need to re load the program by changing motor position.
- Q: How can we observe continuous operation of control action?  
Generate a square wave desired position signal follow this position.

X\_des

NPTEL

PRASANNA S. GANESH | aspb@iitb.ac.in

The slide features a green background with white and yellow text. It includes a gear icon with a lamp inside, a small video inset of a man with glasses, and a graph showing a square wave signal labeled 'X\_des'.

So, now we will see this is a control feedback, so now if what will happen if we keep the desired position constant? Then you imagine their program is start running in their interrupt service routine, you will read the direction or you read the actual position and your desired position is constant and then the actually position will start getting control to the desired position by doing some way.

And then once you do which is desired position it will stop and you will do nothing, or it may reaches close to the desired position and it will stop there only, you start moving it you will find some resistance to move from their position if you externally start moving the motor but otherwise it will do nothing after that.

So, this is not a great way of continuously observing how we are control is performing, you want to kind of have some kind of a handle over seeing that this control what I am implementing is continuously doing something. So, for that purpose we set up the desired position to have like if it reaches a final position then like after some time after steady check is reached you change it back to 0.

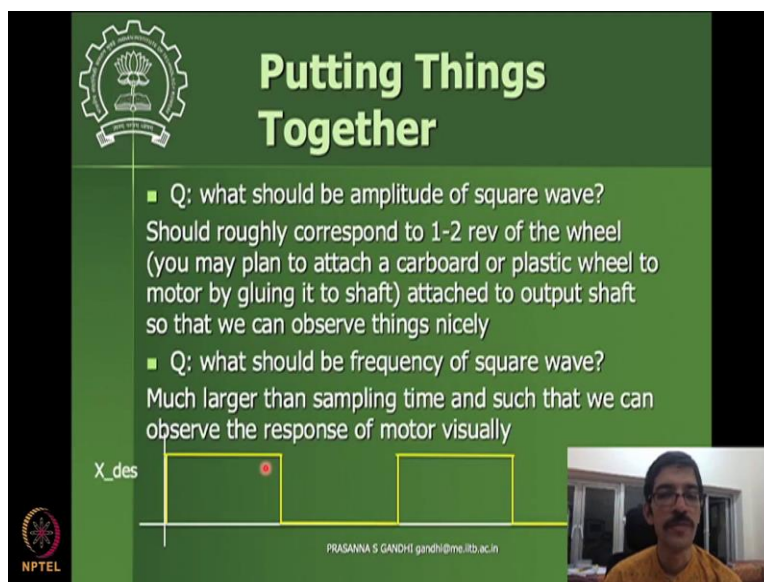
So, you make it 0 again and again you kind of let it come to 0, and again you take it to the desire, so you can do that continuously, so otherwise like if you want to observe the control law working multiple times same kind of control law in multiple times you will not be able to do that. So, you can do that by using the desired position.

Now, how do you generate this kind of a desired position? Think about some algorithm to generate this, so given that you have this now to be written in the interrupt service routine which is kind of a loop which will run continuously every sampling time. So, think about that, suppose if we want the square waveform to have some certain kind of time duration that this desired value will be held constant for this time duration, and then it is brought to 0 for same time duration next, and keep on doing that, with what kind of algorithm comes to your mind, you need to kind of think about and you need to get used to now writing things in interrupt service routine, which is executing every sampling instance.

So, if the sampling time is  $T$  and this time for your square waveform is  $AT1$  then you need to kind of see how many number of sampling instance have to pass by before I change this value, like that we need to kind of these are some of the hints I am giving to for you to write this develop this algorithm for the program.

And this kind of squared wave from you can generate for a desired position, so these desire position, so this is not a tracking, do not confuse that this is becoming tracking know, this because we are suddenly changing this position we are not kind of taking this smooth kind of a trajectory here, it is a multiple times you are observing the regulation here.

(Refer Slide Time: 14:42)



The slide features a green background with a white gear icon in the top left corner. The title "Putting Things Together" is displayed in large, bold, white text. Below the title, there are two bullet points in white text, each followed by a question and an answer. The first bullet point asks "Q: what should be amplitude of square wave?" and the answer is "Should roughly correspond to 1-2 rev of the wheel (you may plan to attach a cardboard or plastic wheel to motor by gluing it to shaft) attached to output shaft so that we can observe things nicely". The second bullet point asks "Q: what should be frequency of square wave?" and the answer is "Much larger than sampling time and such that we can observe the response of motor visually". At the bottom left, there is a graph with a horizontal axis labeled "X\_des" and a vertical axis. The graph shows a square wave with a red dot at the top of the first pulse. In the bottom right corner, there is a small video inset showing a man with glasses and a mustache. The NPTEL logo is in the bottom left corner, and the text "PRASANNA S GANDHI gandhi@me.iitb.ac.in" is at the bottom center.

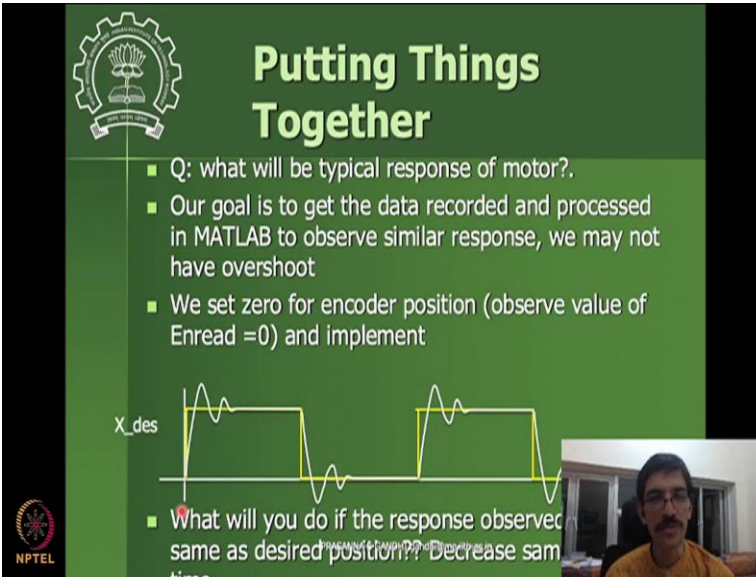
Now, other question, what should be the amplitude of this square wave? So, this now that you can compute based on the encoder accounts that you are coming like know you are getting output



from the encoder. So, say if you are getting whatever 1000 roughly 1000 counts for one revolution then you can plan I want to move 1 revolution or 2 revolution or 90 degree, 80 degree, like that based on that you can plan this number of counts.

And you can attach some kind of a small cardboard or plastic wheel to the motor, maybe you can make it out of the visiting cards there kind of stiff stuff there to attach and you can put a pointer there to kind of like you know the position, so like that you can do. Then what should be frequency? Frequency should be large enough, so that you observe the controls so your control is happening like that, it is controlling and then it is becoming steady state, there should be some duration left beyond that and then it should be change. So, this typically frequency of the square wave should be very very low, so that you are able to observe your control nicely.

(Refer Slide Time: 16:07)



The slide features a green background with a white gear icon in the top left corner. The title "Putting Things Together" is written in a bold, white font. Below the title, there are three bullet points in white text. The first bullet point asks a question about the motor's response. The second bullet point states the goal of recording and processing data in MATLAB. The third bullet point describes setting the encoder position to zero. Below the text, there is a graph with a yellow square wave labeled "X\_des" and a blue oscillating line representing the motor's response. A small inset video of a man is visible in the bottom right corner of the slide.

- Q: what will be typical response of motor?
- Our goal is to get the data recorded and processed in MATLAB to observe similar response, we may not have overshoot
- We set zero for encoder position (observe value of Enread =0) and implement

X<sub>des</sub>

- What will you do if the response observed same as desired position?? Decrease sam time

So, you can observe this response visually, what should be what will be typical response of the motor, if you can sketch on this desired response, what is the actual response going to be? You can get something of this sort. So, you can look for a you can get some response of this kind when you give this square wave form. And then we want to kind of get this data out of the microcontroller and maybe we should be able to put it in the MATLAB to observe this response, this response we can observe by graphing functionality to see in real time in the coming at the graph.

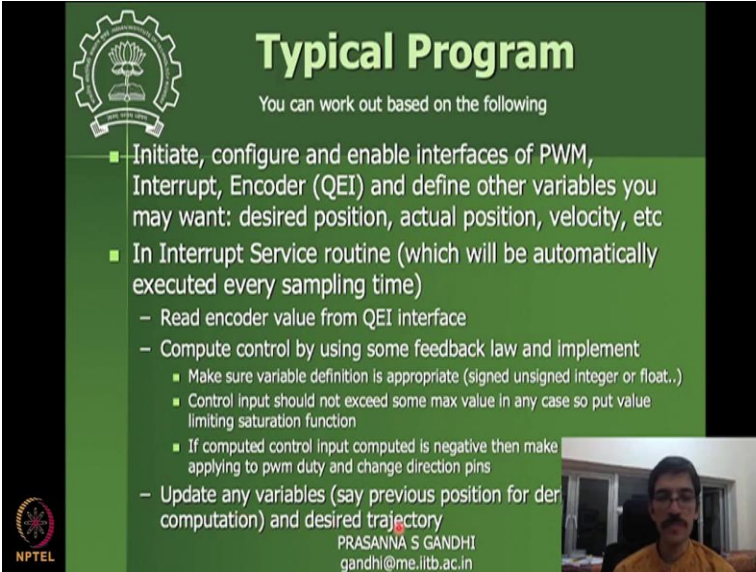


So, of course we need to initialize some value to be 0 and proceed. So, you may it may so happen that you will find that your response observe is exactly same as the desired position, you may not observe any undulations here, especially if you are observing for just speed of the motor, no open loop even some kind of a you know it is desired it is like a PWM which is given some duty cycle.

Not PWM, but speed sorry this is like you know speed that will desired, PWM desired will give you position continuously going in one direction till the time this PWM is get into 0. But what I mean is here if you have a constant voltage given with the, to the motor and we are observing this  $\dot{x}$  dot desire then that response is going to be like that, it will go like a first order kind of function.

But if your inertia is too low then the sampling time that you have chosen might be too large or capturing the initial kind of a you know that typical first order kind of response capturing may not happen, so you may find that okay oh my waveform for the speed now is coming as a straight, so then you need to decrease that time sampling time to make sure that you are able to observe that transient that is going there. These are kind of small small tricks that you will need to learn as you build your practical understanding of the system.

(Refer Slide Time: 19:01)



**Typical Program**  
You can work out based on the following

- Initiate, configure and enable interfaces of PWM, Interrupt, Encoder (QEI) and define other variables you may want: desired position, actual position, velocity, etc
- In Interrupt Service routine (which will be automatically executed every sampling time)
  - Read encoder value from QEI interface
  - Compute control by using some feedback law and implement
    - Make sure variable definition is appropriate (signed unsigned integer or float..)
    - Control input should not exceed some max value in any case so put value limiting saturation function
    - If computed control input computed is negative then make applying to pwm duty and change direction pins
  - Update any variables (say previous position for der computation) and desired trajectory

NPTEL  
PRASANNA S GANDHI  
gandhi@me.iitb.ac.in

So, this typical program you can use this following what you say hints to develop on your own. So, you need to initiate configure and enable interfaces, all the interfaces that you are needing for

the for this closed loop system operation to be working, PWM, interrupt, then encoder QEI module and any other variables you want to define you can define them to begin with, this all can happen either in the main program or in the like you know say it's some of the definitions can happen before main program, when you use the commands like include something then thing like that.

Then in interrupt service routine this is very important, we will need to do these three operations that I was talking about. So, in these you need to make sure no initiation or configuration kind of functionalities from QEI should come into this interrupt service routine, configuration are enabling kind of interfaces, they should not come in the interrupt service routine, it should have just for example from QEI interface it should just read reading the value of the encoder whatever function is there, reading the speed of encoder whatever function is there that can come up here.

Now, that is a thing that is in the continuously changing, all other things are going to be for constant for this particular program. So, only the changing parts should come which are changing every interrupt cycle that should come. Now, then you have to compute control by using the feedback law, you may have seen some law proportional law or something like that and implement.

Now, implementation aspects will involve a couple of things here, that we need to be sensitive to say for first make sure that the variable definitions are appropriate, your signed and unsigned integers or floats or whatever these definitions are made in a way that it is conducive, for example if you define for the control input the unsigned integer then like your negative part of the program control will not work at all.

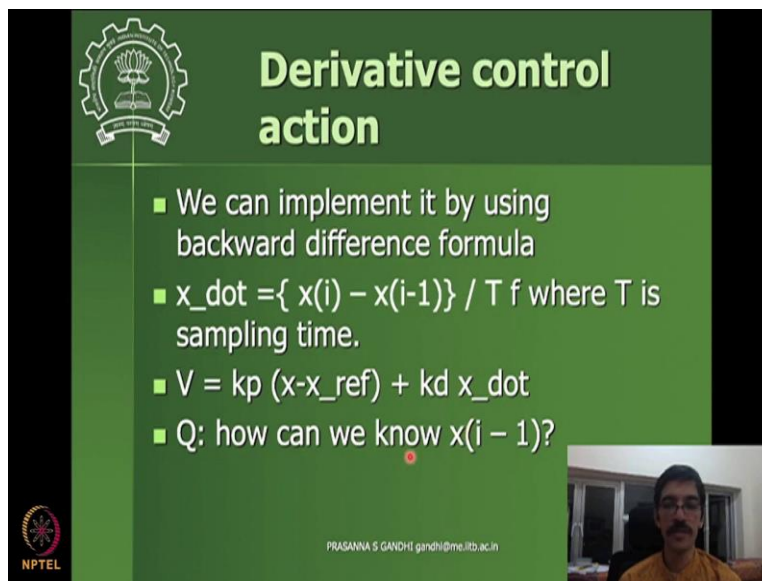
So, those kind of things you should not be you should be careful about. Then control input should not exceed some maximum value in any case. So, put the value with a in some kind of a limiting saturation function, and then third part is like if computer control computed is negative then how do you kind of handle that? What is mean it means that okay I need to reverse the direction of the motor with some same so you will make it positive before applying to PWM duty.

So, PWM duty is like mod part of the control input command and the sign part like you know positive or negative will be used to change the direction pins. This is a very important kind of

baseline for like control implementation to happen. These two points are coming so you can maybe kind of put some kind of a small function to make sure that this will happen before control is implemented.

Then you need to update any variables. So, previous position needs to be updated for derivative computation included that, or desired trajectory need to be updated now, so that functions will come here or they can come in the start also, somewhere they should come in the interrupt service routine.

(Refer Slide Time: 22:48)



The slide features a green background with a white gear icon in the top left corner. The title 'Derivative control action' is written in white. Below the title, there are four bullet points in white text. In the bottom right corner, there is a small video inset showing a man with a mustache and glasses. The NPTEL logo is in the bottom left, and the email address 'PRASANNA S GANDHI gandhi@me.iitb.ac.in' is at the bottom center.

**Derivative control action**

- We can implement it by using backward difference formula
- $x_{\text{dot}} = \{ x(i) - x(i-1) \} / T$  where T is sampling time.
- $V = k_p (x - x_{\text{ref}}) + k_d x_{\text{dot}}$
- Q: how can we know  $x(i - 1)$ ?

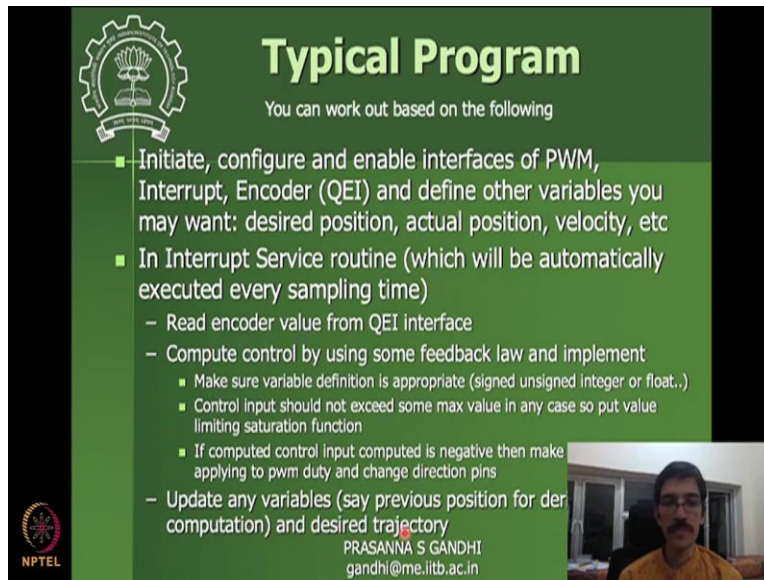
NPTEL

PRASANNA S GANDHI gandhi@me.iitb.ac.in

So, how do you compute derivative control action? Do you know this difference formula you can use that and do that. So, it is very straightforward kind of a thing only thing when the question is how do you get this previous, how do you know what is previous x position, position x, I know current position but how do I know the previous position? Since we do not in the interrupt service routine when we are working with then we need to store the previous values.

So, we store the current values which will be available as x previous for the next sampling instance to come. So, that is how we store the previous value every, in every that is what every sampling instance. So, we update these values to kind of make sure or previous values are available.

(Refer Slide Time: 23:42)



**Typical Program**

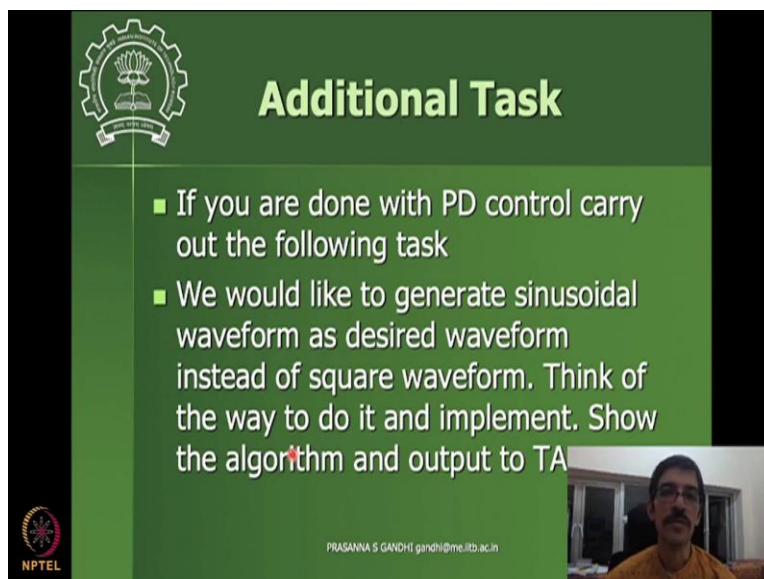
You can work out based on the following

- Initiate, configure and enable interfaces of PWM, Interrupt, Encoder (QEI) and define other variables you may want: desired position, actual position, velocity, etc
- In Interrupt Service routine (which will be automatically executed every sampling time)
  - Read encoder value from QEI interface
  - Compute control by using some feedback law and implement
    - Make sure variable definition is appropriate (signed unsigned integer or float..)
    - Control input should not exceed some max value in any case so put value limiting saturation function
    - If computed control input computed is negative then make applying to pwm duty and change direction pins
  - Update any variables (say previous position for der computation) and desired trajectory

NPTEL  
PRASANNA S GANDHI  
gandhi@me.iitb.ac.in

As I said here update any variables, because variables need to be getting updated, so  $x$  previous will need to get updated to current  $x$  current to make sure that it is now the current value will become previous value for the next sampling instance, that is a kind of thinking philosophy that one should use. So, you want one or two or three previous values, you can do that very same way.

(Refer Slide Time: 24:08)



**Additional Task**

- If you are done with PD control carry out the following task
- We would like to generate sinusoidal waveform as desired waveform instead of square waveform. Think of the way to do it and implement. Show the algorithm and output to TA

NPTEL  
PRASANNA S GANDHI gandhi@me.iitb.ac.in

And so, this is some additional tasks you can think about say if you want to generate sinusoidal waveform as a desired waveform, instead of the square waveform that I have like you know you

have seen here, how do you kind of do it and implement in the interrupt service routine? So, this is algorithm one can think about.

So, this is just extra tasks that you can think about? So, I think this is good enough kind of a primer for you to kind of get started and implement practically any kind of control on your microcontroller. And then we will have an interesting kind of things to talk about with the practical results that you get out of such implementation. So, we will be able to talk about in your assignments as we progress. So, maybe we will stop here for now and then have some more part of the lecture coming up.