Design of Mechatronic Systems
Professor Prasanna S. Gandhi
Department of Mechanical Engineering
Indian Institute of Technology, Bombay
Lecture 20
Hardware Interfaces

(Refer Slide Time: 00:11)

So, in the last class, we were looking at programming philosophy, so of microcontroller in general. So, microprocessor has different-different kinds of way interfaces available and we saw some general philosophy of programming interfaces also in the last part where we looked at control registers and actually working registers.

So, now in this class we will look at different kinds of interfaces that are available and typically with some examples, we will look at their little more detail about what is their utility, how they can be used for useful applications.

(Refer Slide Time: 01:23)

So, let us begin different kinds of interfaces possible. As we can see here, so, first is digital input output interface. The digital input output is most common interface that is there in all microprocessors or microcontrollers. So, this interface allows you to exchange the data in the digital format either input or output with the external peripherals.

Say, for example, if you have indicators or some LED light that you need to glow after some process has happened or indicating something has happened, then you can use these digital interfaces. Then you can have limit switches to, in the operation you want to kind of control the application such that it does not damage the system. So, when this limit switch is reached that is the indicator that your operation beyond this point is going to be harmful.

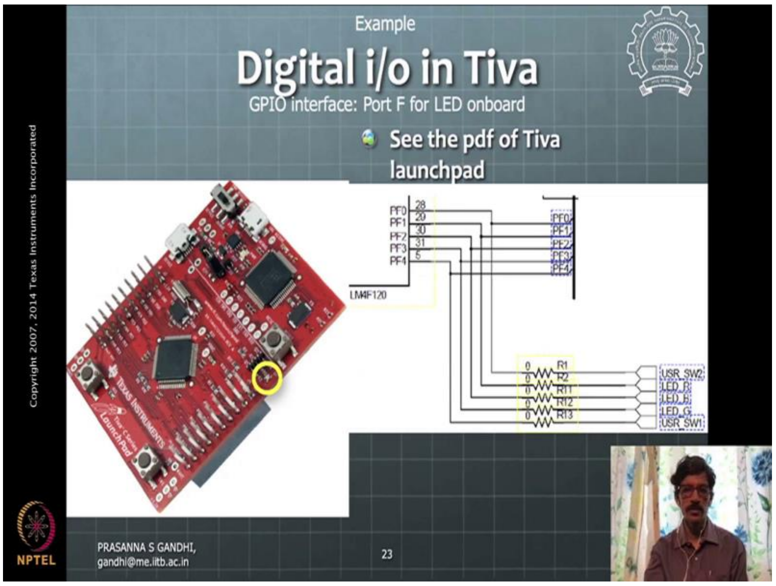
So, when that switch is reached, we will switch the system off or do something to save the system. Like that, you can use these digital input-output interfaces. Other very important application for this is stepper motor running. There are different interfaces available for stepper motor running nowadays, you can use digital input output as a most basic kind of interface with amplifier or you can also use PWM interface nowadays for this.

Then you can use this for direction control in the servo motor control. So, there are ample applications and this is most important interface to be used. So, say for example, we saw 7 segment displays in our previous classes. So, to program that 7 segments display you will need this digital input output interface by using that you can display some numbers or some data to the user.

So, how this interface works is very simple actually, you know the registers now, the registers, if the pins of the reducers are made available to the user, normally they are not available to the user. So, given a register if the pins of the register are made available to the user, then user can use that those pins for where the data is high or low that depending upon if the value in the register is 0 or 1, the pin will go high or low and that is what is indicated and used by the user in some way.

And you can have some logic to have the facility for this wire to input the data to the register as well. Then, reading and writing to this digital input output is done by using like some control registers by programming control registers first, where you define whether the given register or digital input output port is so to say it is called is used in the input mode or output mode. So, for that you need some control registers typically.

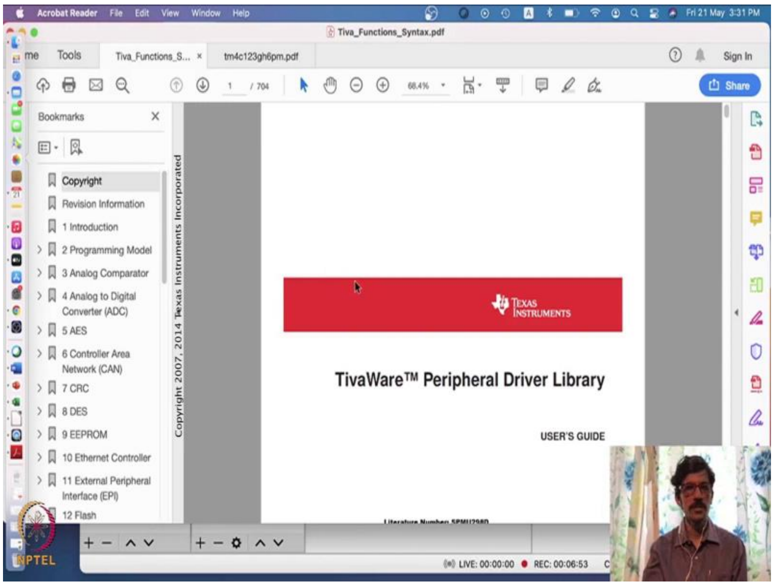
(Refer Slide Time: 04:48)

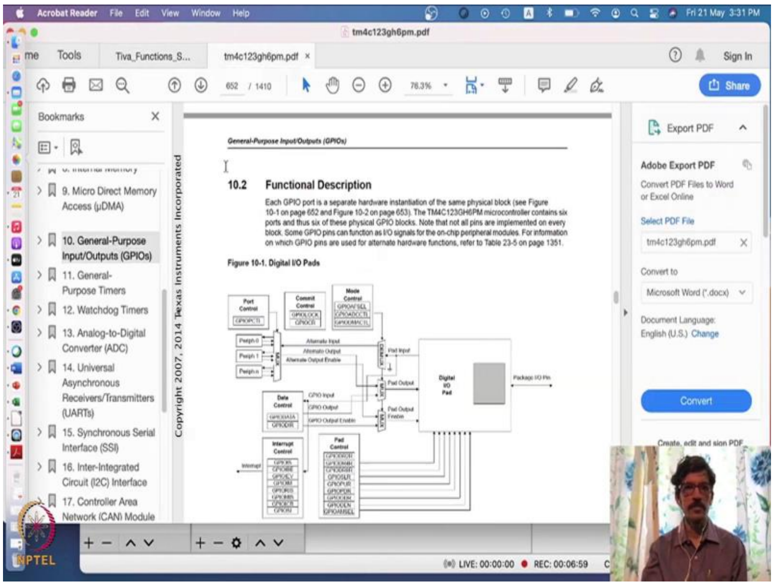


Now, let us have a look at little bit of more details about this Tiva launchpad that is what we will is optional but we will use in this course as a hardware for this Tiva interface, this Tiva board that is picture is seen here the F port for example in Tiva is used for LEDs online some part of the LEDs, there are so these 3-4 LEDs that are there on the board there are 3 LEDs actually.

So, R G and B these 3 LEDs are there and they are connected to some Tiva port. Connection pin diagram is shown in this Tiva datasheet. Now, the way this Tiva input output pins, other pins this is F port but there are some other ports available on these pins. And interestingly, this is very compact and more kind of a modern ARM architecture as we saw based interface or microcontroller interface for this digital input outputs. So, let us have a look at little bit more details about what facility this interface gives in Tiva.

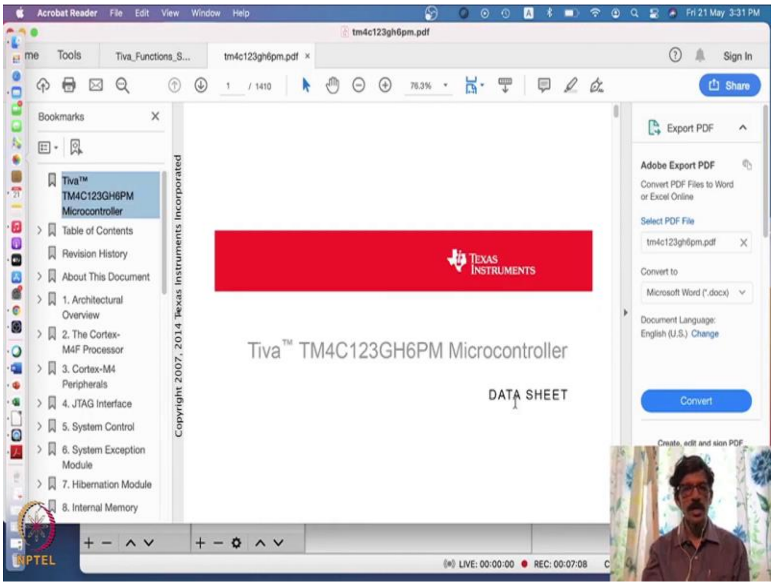
(Refer Slide Time: 06:17)

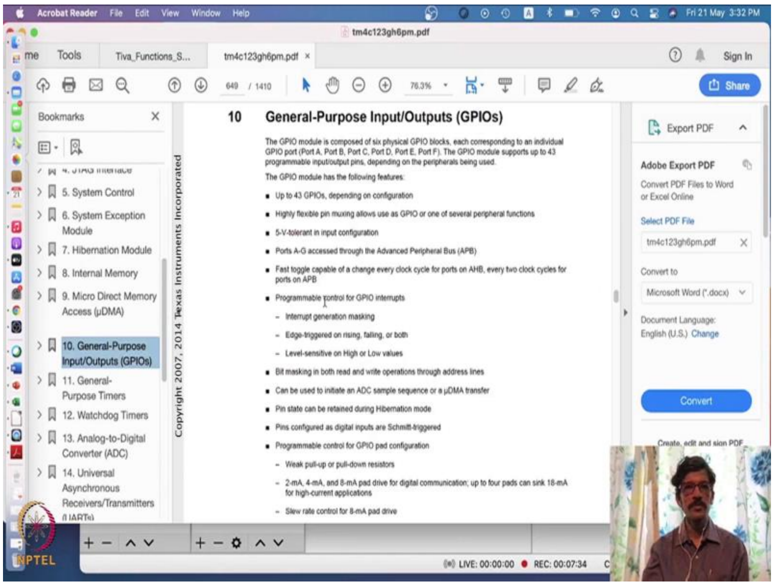


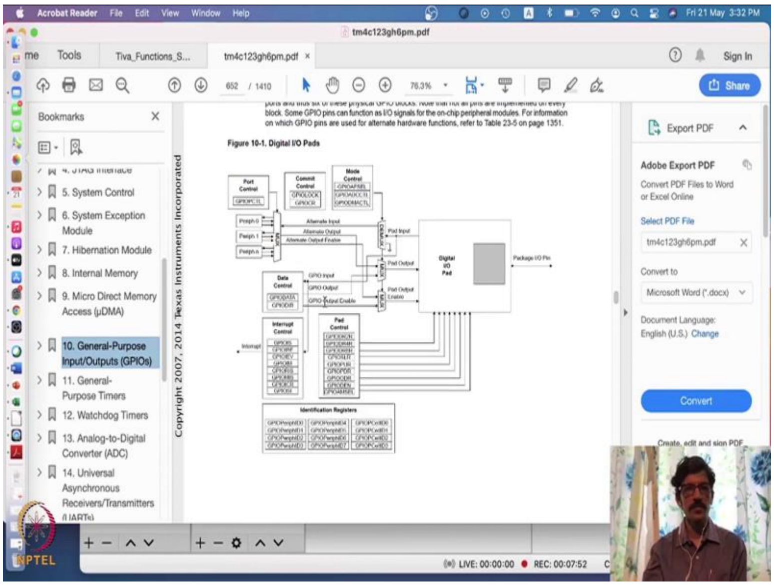


So, before going to the ADC, we will look at the Tiva in little more details here. So, you can see that the Tiva peripheral driver library, this particular file PDF file will give you all the details about the functionalities or the API library is there. So, this is the driver library, API functions are there, this function description is given in this file and in these you have the, this is a Tiva datasheet for the Tiva.

(Refer Slide Time: 06:55)





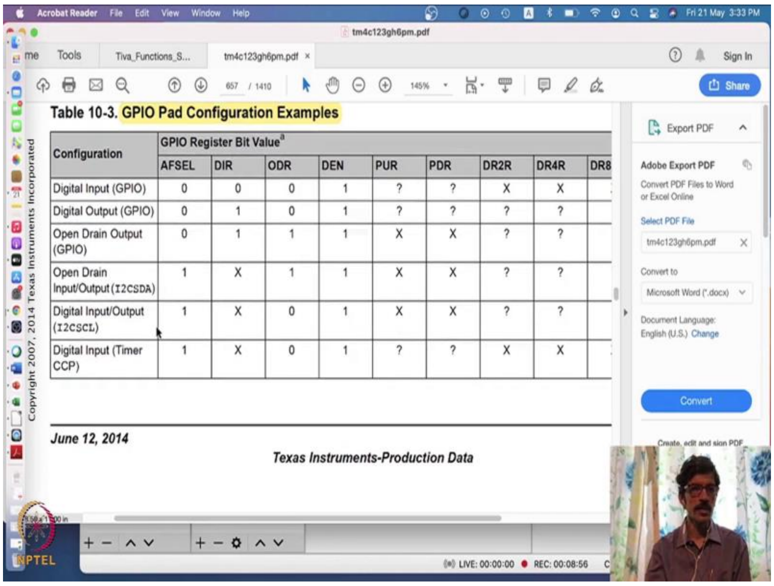


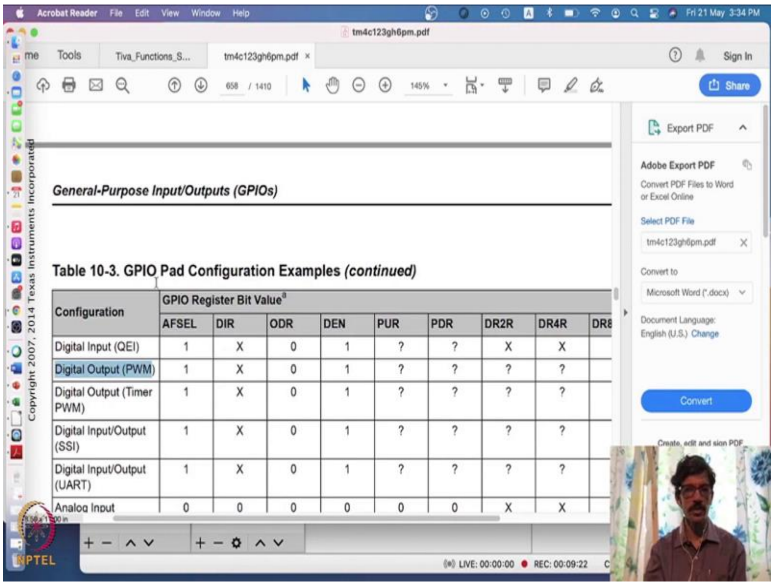
So, this is microcontroller datasheet and this datasheet also gives you the details of the GPIO interface. So, depending upon this programming philosophy, you will use either of these two infer for the programming. Say, for example, if you use the Tiva microcontroller data sheet, it will have these all these different, different registers that are defined here.

So, let us say GPIO interface if you see here general-purpose input output. So, it will give you the different-different registers that are used in GPIO interface, for example. And some kind of a block diagram of the architecture of this general-purpose input output interface and so on and so forth all these registers where you can use for programming and controller that will be given here.

So, in this you will have directly functionality given. So, if you are using directly API functions of this driver library, then you do not need to worry about these different-different registers that are control registers or working registers, you need to understand how these functions are written.

(Refer Slide Time: 08:21)

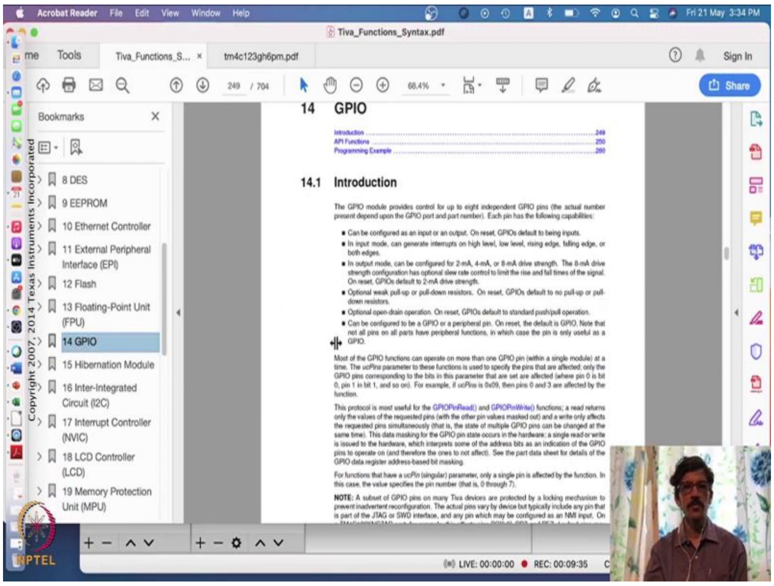


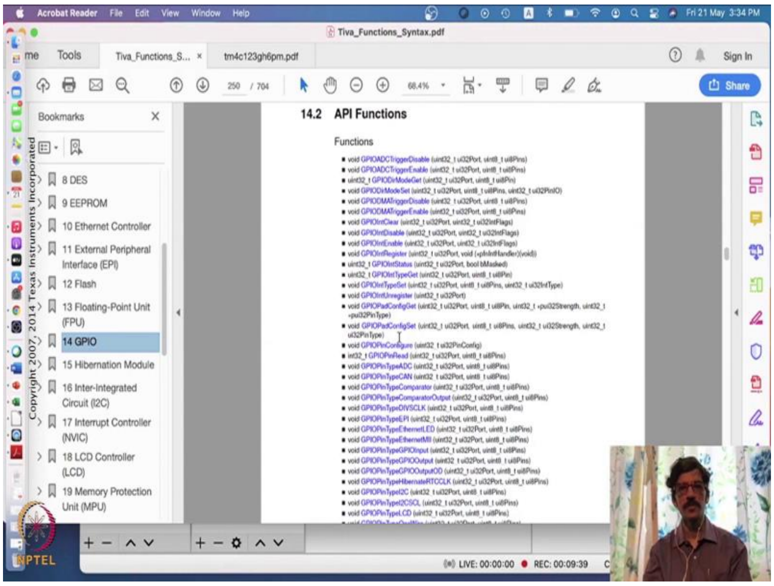


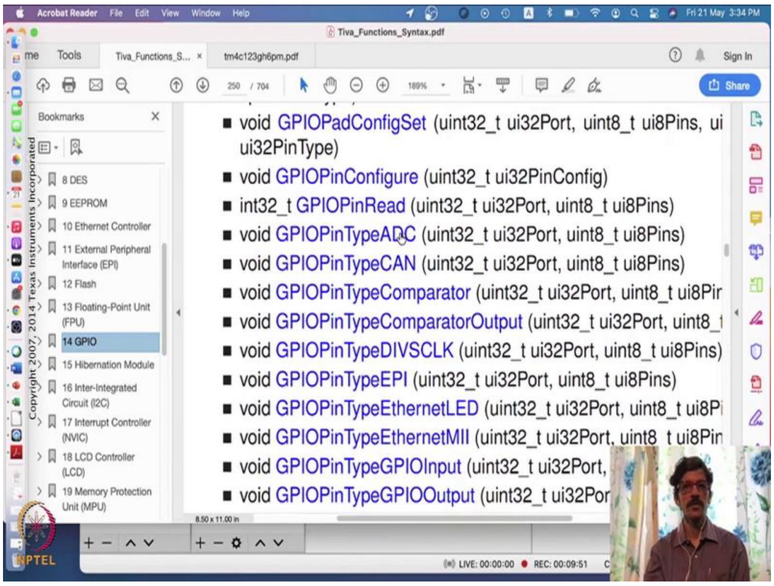
And interestingly, if you see this Tiva general- purpose input output interface, it has this port this general-purpose input output pins have mapping possible. So, for example, these pins can be like normal function or alternate function selection is there, so where you can select it as a say for example, QEI interface pin. So, this, for these digital pins you can configure this pins to work as a quadrature encoder interface or they can work as a PWM pins.

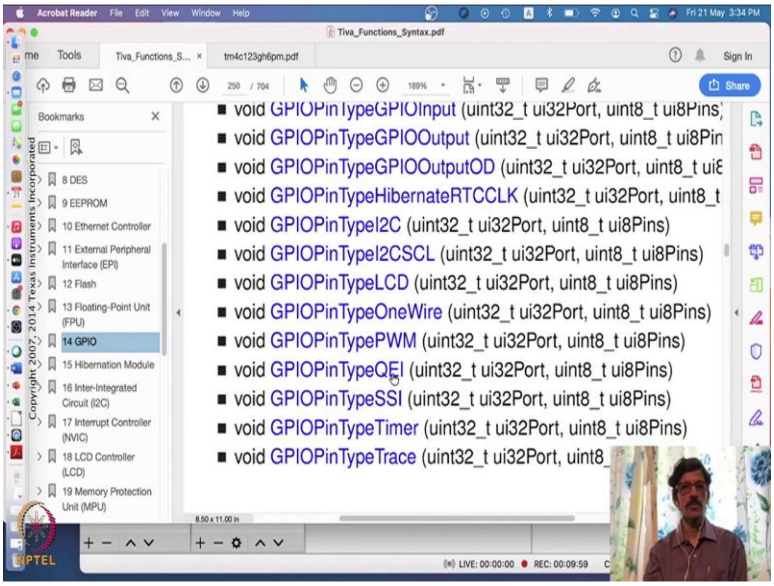
So, this configuration is done via two ways one is by using these control registers that are given in this datasheet of the Tiva, the chip or the functions.

(Refer Slide Time: 09:24)




14.2 API Functions
Functions

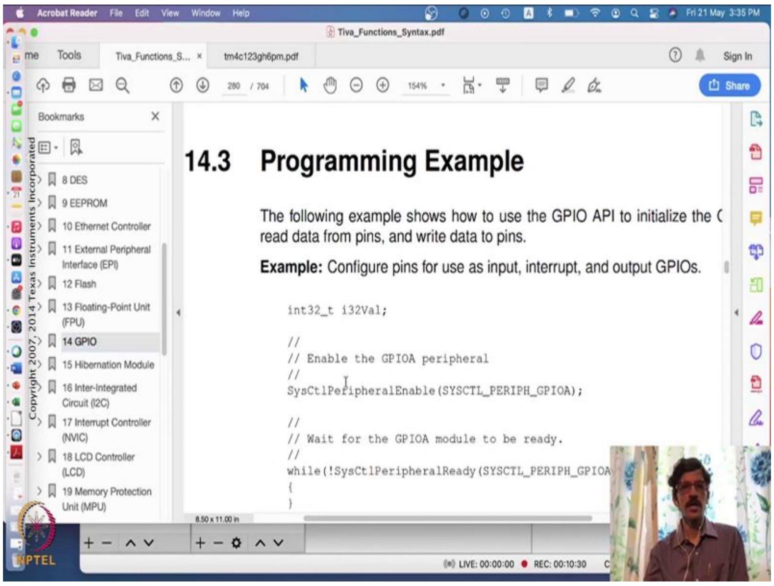

void GPIOPadConfigSet (uint32_t ui32Port, uint8_t ui8Pins, ui
ui32PinType)
void GPIOPinConfigure (uint32_t ui32PinConfig)
int32_t GPIOPinRead (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeADC (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeCAN (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeComparator (uint32_t ui32Port, uint8_t ui8Pir
void GPIOPinTypeComparatorOutput (uint32_t ui32Port, uint8_t
void GPIOPinTypeDIVSCLK (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeEPI (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeEthernetLED (uint32_t ui32Port, uint8_t ui8Pi
void GPIOPinTypeEthernetMII (uint32_t ui32Port, uint8_t ui8Pin
void GPIOPinTypeGPIOInput (uint32_t ui32Port,
void GPIOPinTypeGPIOOutput (uint32_t ui32Por

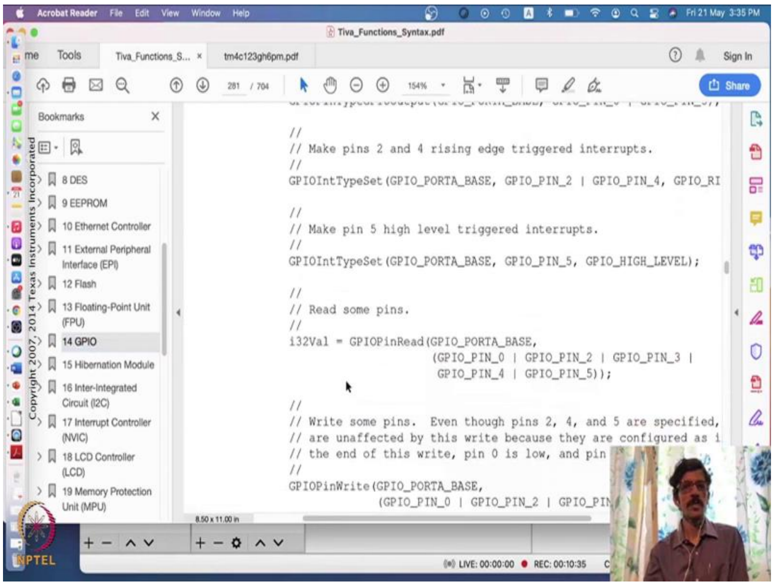

void GPIOPinTypeGPIOInput (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeGPIOOutput (uint32_t ui32Port, uint8_t ui8Pin
void GPIOPinTypeGPIOOutputOD (uint32_t ui32Port, uint8_t ui8
void GPIOPinTypeHibernateRTCCLK (uint32_t ui32Port, uint8_t
void GPIOPinTypeI2C (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeI2CSCL (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeLCD (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeOneWire (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypePWM (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeQEI (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeSSI (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeTimer (uint32_t ui32Port, uint8_t ui8Pins)
void GPIOPinTypeTrace (uint32_t ui32Port, uint8

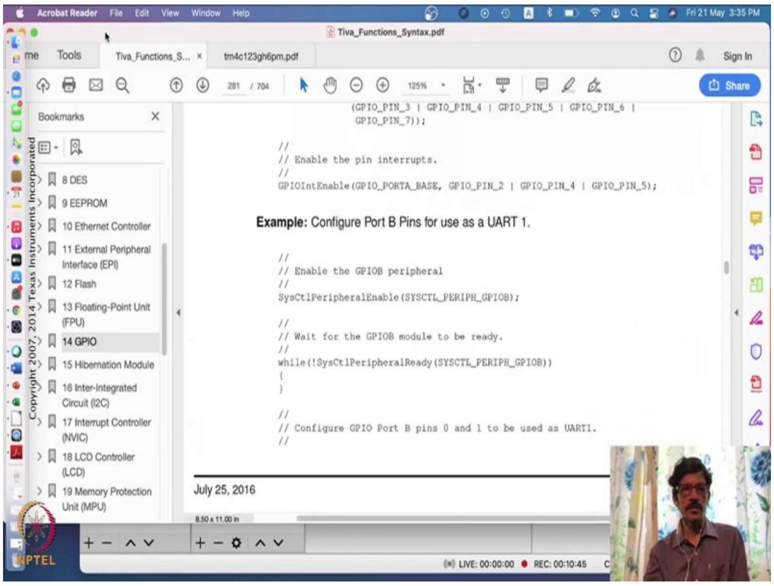
So, for example, if you go to these GPIO functions, then it has some introductions and there are different-different functions that are given and some of these functions are having this pin configuration if you see here is a pin configure kind of a thing. So, you configure this pin for different-different types of pins, it can be as an ADC pin or it can be as general- purpose input output pin or PWM pin or QEI pin like that.

So, this function you can directly use then for the configuration. So, like that you start reading through the datasheet understanding these different-different functions. So, if you go to details of this function, you will find a way to use it and write it.

(Refer Slide Time: 10:12)

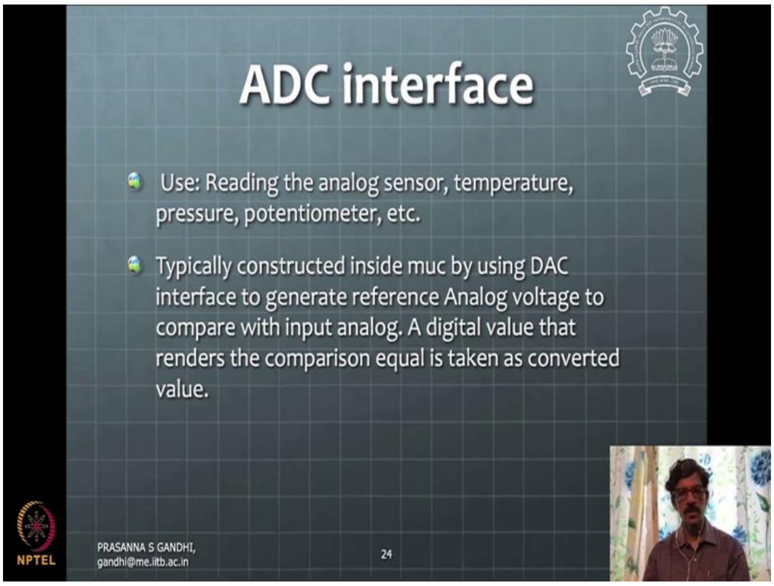






And then there are these of course, there are these programming examples that are given these are functions here list of functions and programming examples and then you can directly go to programming examples and see these sample programs and figure out what changes you need to make to do it for your own purpose. So, let us get back here.

(Refer Slide Time: 10:39)



So, that is what about GPIO, general-purpose input output interface. So, in this particular case of Tiva that the interface has possibility of many different functions that can be possible for that interface. So, you can program the pins. So, pins are programmed that interface is GPIO, but the pin can be programmed to have different-different kinds of other interfaces possible there, which may not be there in all the microcontrollers.

This modern microcontroller which is ARM based microcontroller, Tiva has that functionality or some microcontrollers XEP 100 for example, will have only input output pins, they are non-configurable, you cannot change their functionality to use them as a PWM pins. So, depending upon what is written in the datasheet and what is of your interest, you can start using them.

Now, let us look at this ADC interface. So, this as the name says analog to digital conversion interface and it is used for reading analog sensor. So, you have analog sensors like temperature sensor, pressure sensor or potentiometer, accelerometers, they have analog output and that analog output needs to be sensed or you want to say some like resistor value resistance, resistor is there in which when the current changes you want to sensor voltage there, those are kind of places you use these ADC sensors.

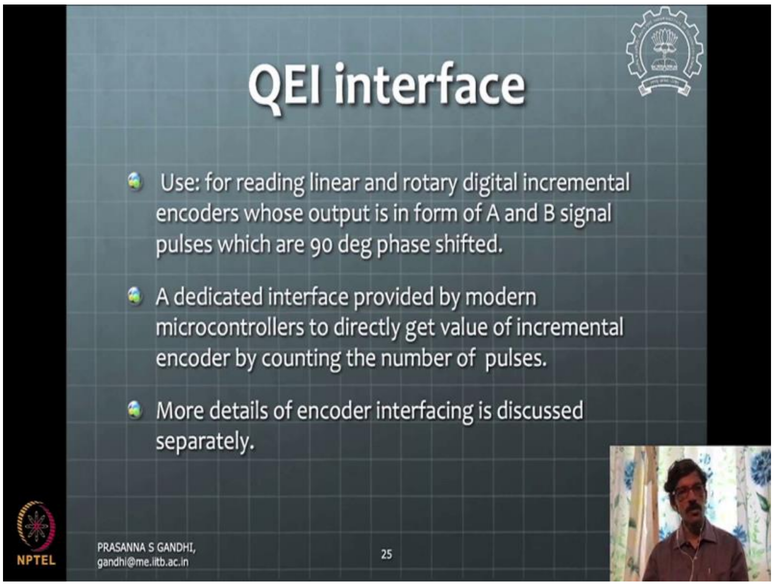
You have the strain gauges, for example. So, there are many, many different utilities for this analog to digital conversion. And how these analog to digital conversion happens is by using a comparator you have this digital to analog conversion interfaced first, which uses digital value given by microcontroller and processes it to give analog output.

How does it do that? Maybe, we have something to show I will show you some stuff later. So, this digital analog conversion is done first and analog value is available. Now this analog value on one of the pins that is available is compared by using op-amp comparator with the analog value that user wants to sense.

And when these two values match each other, then the digital value that is supplied to this pin, which is used for comparison that digital value is going to be the value for analog to digital conversion. So, this is how the analog to digital conversion works by using the comparison with value that is generated by the microprocessor itself through digital to analog conversion.

So, this interface is also very popular we need to use it many places in mechatronic systems for various kind of sensing application. and this is IoT, for example, is very much in IoT this is of great value, this interface. A lot of values can be sensed analog like temperature or acceleration that in the places can be sensed and that can be used in the IoT way to monitor some situations around any machine.

(Refer Slide Time: 14:27)



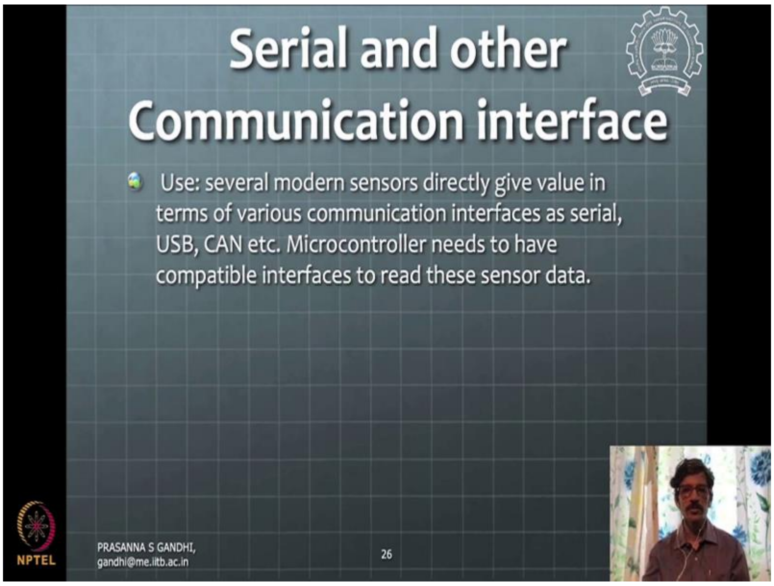
Then, this is a very important other interface called quadrature encoder interface. This interface basically is used for encoders. So, we have seen in the sensors lecture, what are the encoders. So, those encoders view directly want to interface with microcontroller you will use this quadrature encoder interface. This has, as we saw before, it has a signal of the form A and B where both are pulses which are 90-degree phase shifted those A and B are taken as a input to this quadrature encoder interface and then they are processed to generate a number or they are counted to get a number, which will correspond to the position of the encoder.

So, this is nowadays very useful interface because encoders are digital sensors for position instead of using your potentiometers, encoders would be preferred because they will have absolutely no noise because they are digital in nature and that has a lot of important applications in position control systems.

So, a lot of mechatronics positions, positioning system, positioning stages would use encoders and for those encoders this interface will be very useful. So, printers or machining systems, CNC and others, they will have these encoders. Then we will discuss this more details about this interface and how to program for particularly for Tiva we will discuss a little bit more detail in another class.

And that can be used for programming these hands on training your own Tiva microcontroller for using this interface along with the motor that is there in the kit.

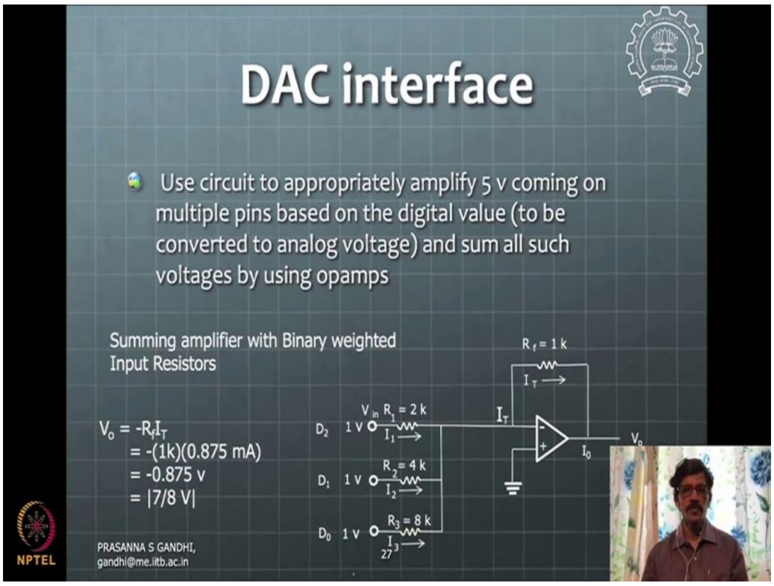
(Refer Slide Time: 16:36)



Then you can have serial and other communication interfaces, which are nowadays available like a USB communication or CAN communication. So, this control area network communication interface, that has been specifically designed for automotive control systems.

So, these are all different-different kinds of interfaces that can be possible for serial and other communication. You can have any some Wi-Fi interface or wireless communication interfaces also are possible. So, although in Tiva it is not there, but there are other kind of microcontrollers, where you can find the direct inbuilt interfaces or with Tiva if you want to use you can use ZigBee wireless devices to communicate with.

(Refer Slide Time: 17:37)

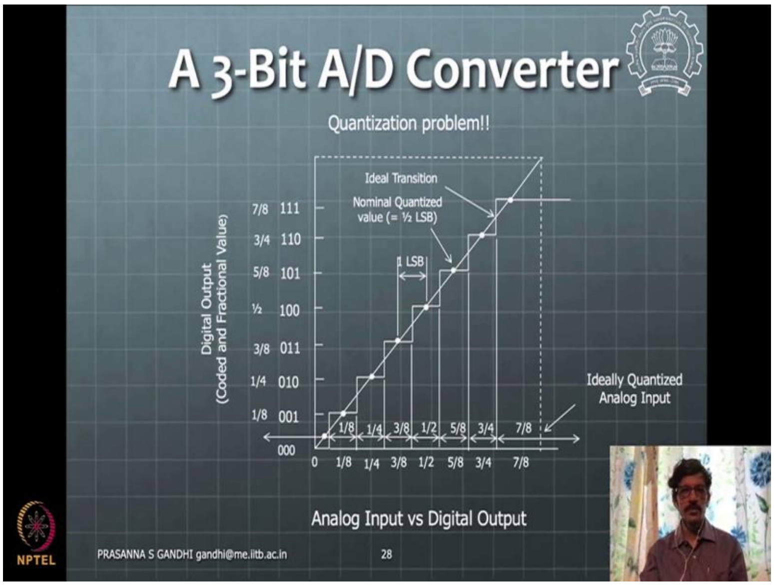


Then, so, this is where like I was talking about this digital analog conversion interface, which is there in the analog to digital conversion interface. So, to make ADC work you need to have a DAC inside ADC. So, this DAC interface can be also used separately. So, typically this interface would work very simply you can have this circuit, which is some op-amp with these dividing resistors those provide this current here and then this current is amplified and you produce this analog voltage output.

And this current is proportional to the digital values that you are supplying at every, each of these resistors and these resistor values are so selected such that the calculation will show that your, when this digital value is changed linearly, analog output voltage will change linearly. So, this is summing op-amp which will sum up all these different-different voltages which are coming here because of the differential currents that are proportional to the digital value that you provide here.

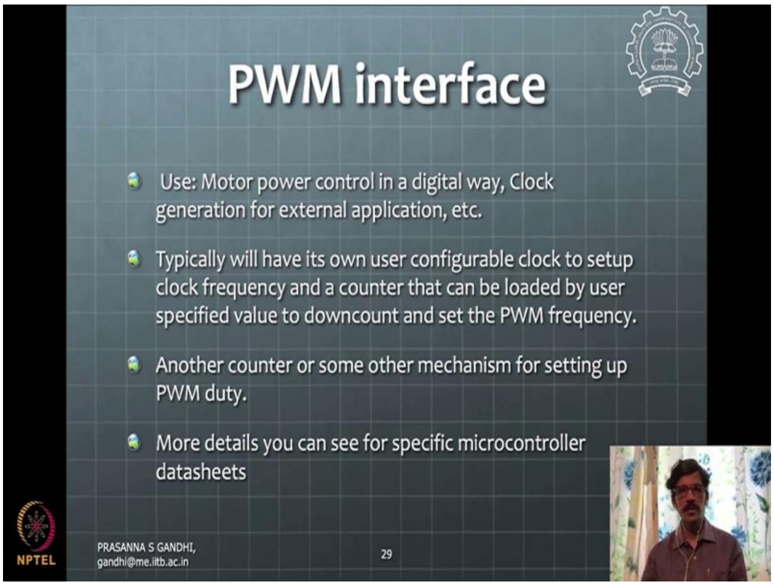
And higher the digit says a D 0 is the least significance digit, its registers for resistance will be very high and the most significant will have a low resistance so that it will get more representation in the summation. So, this is typically the digital to analog interface works. And this is not very wide, but some applications where you want to run you have a motor which you want to run and it is having an amplifier, which is analog to analog conversion amplifier there you need digital to analog interface. Otherwise, typically for running motor we use PWM interface, which we will see.

(Refer Slide Time: 19:54)



The main characteristics of the digital output is that it will have the stepping behaviour of course, because your voltages or your digital values are going to be in discrete form and depending upon the number of bits that you use for this conversion so, you have a 3 bit conversion, then the least count will be 1 over 8, you will have that much error in conversion like that you have this possibility. So, this is called quantization errors typically that exist in these systems.

(Refer Slide Time: 20:44)

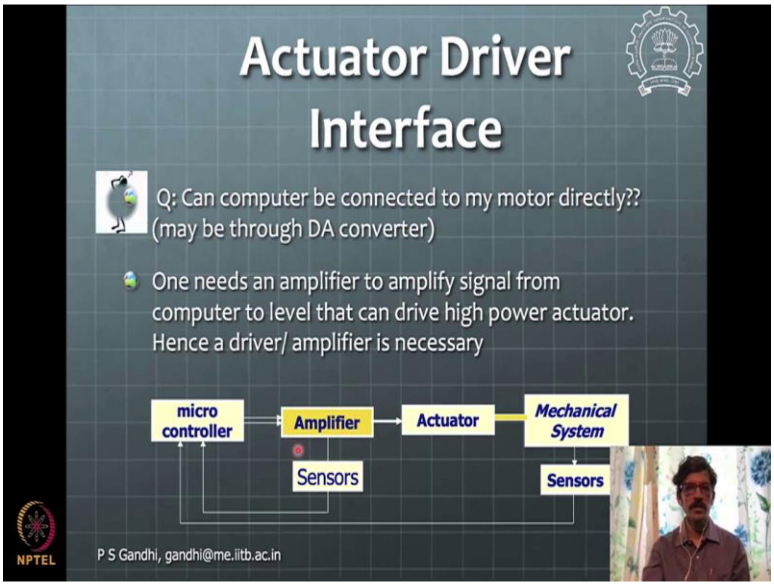


Now, we come to this PWM interface, which stands for pulse width modulation interface and this is very important interface for mechatronics engineers because this interface you can use for driving many different kinds of actuators or you can use it for cloud generation application for external peripherals. There are many different ways this interface can be used, but the most important is driving the actuators by this interface, why these interfaces used for driving actuators we will see that.

So, how this interface works is basically it uses a clock signal of microcontroller and divides that or uses that to generate some pulses. Now, let us first understand what is this pulse width modulation. So, this pulse width modulation will be where you will have a pulse coming on the pin and that at some frequencies say maybe about a kilohertz or 4 kilohertz frequency. The pulse comes and the pulse has typically clock pulse will have 50 % duty means the clock pulse will have 50 % on time and 50 % off time.

PWM you can have this different, different possibility of wearing that duty in the sense you can say this is a 10 % PWM, 10 % duty PWM where the on time can be changed depending upon what is application or what is the requirement. So, this is PWM interface provides a facility to change the frequency and on time for the pulse. So, we will see, more programming details will be there in the microprocessor datasheet, microcontroller datasheet and we will have a look at in the further classes.

(Refer Slide Time: 22:51)



So, this now let us look at how this motor, why motor needs PWM interface. So, if you want to drive an actuator so, you need a driver for amplification. So, say because you cannot say I will connect my motor directly to my microcontroller and it will start running. No, imagine you cannot connect the ceiling fan, for example, to your microcontroller and expect it to run because the typically these motors will require larger power.

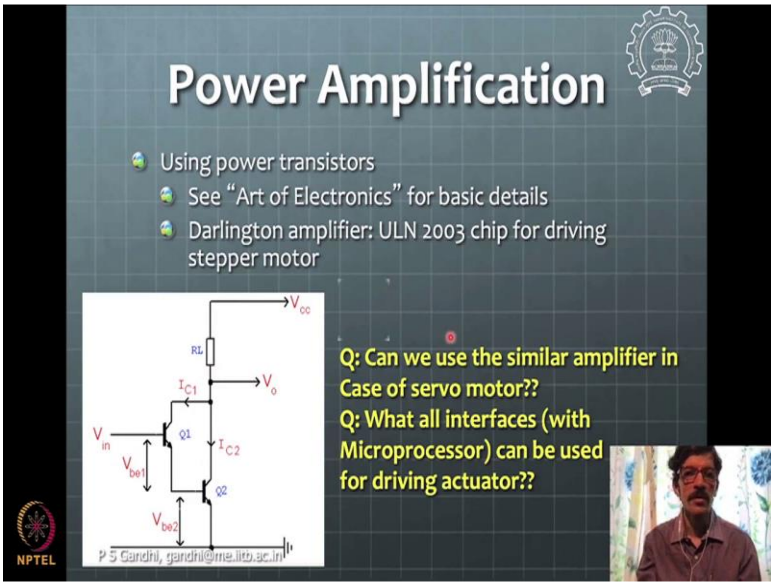
Microcontroller are supposed to be very low power devices they will provide hardly any current for these applications to run. So, what we do is we amplify the current that we or current or voltage whatever you want to say that we get output from the, our microcontroller and then provide it to the actuator.

And suppose let us say if we use some amplification here. So, if we use say some transistor-based amplifier for example. So, typically amplifier is required. So, this will be the configuration that you will use. So, microcontroller output goes to amplifier, amplifier is the thing increases power and give it to actuator and then actuator drives your mechanical system.

And the sensors will sense the feedback and it will provide that feedback to microcontroller and microcontroller will take some decision to process this and give power to the amplifier. This is a typical loop in which the microcontroller, your mechatronic system will be working.

Now, so, we are focusing on now this amplifier unit here. So, what kind of amplifiers we will use?

(Refer Slide Time: 25:06)

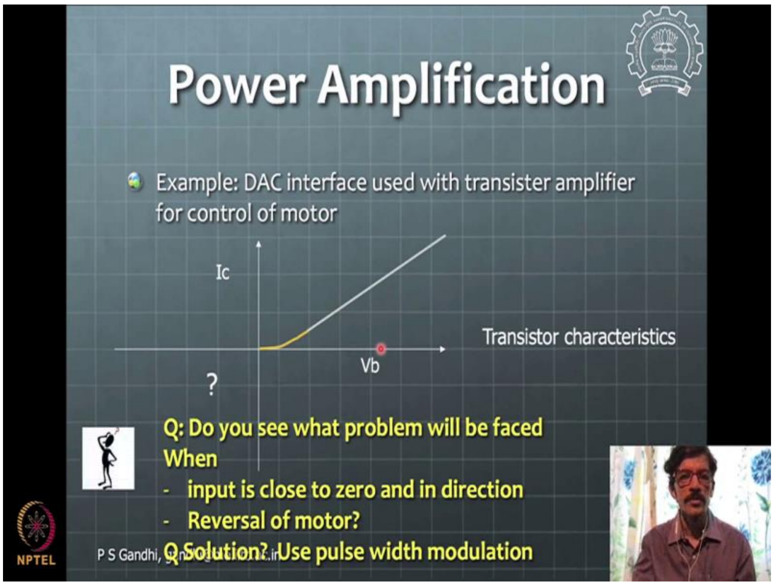


Suppose, we have a transistor-based amplifier. So, say ULN 2003 chip for driving stepper motor is there like that we are using some transistor amplifier. So, if we use this amplifier in the servo motor, so what we are talking here is right now, when we are talking in this actuator driver interface, we are typically the servo motor we are talking here not a stepper motor here and this servo motor not really this aero modelling servo motor.

So, there are different kinds, there is a different, although the terminology is same what we are talking about is DC, permanent magnet DC motor we will use this term permanent magnet DC motor than a servo motor. So, when we have to drive permanent magnet DC motor, then if we use this circuitry and then here, (we input) we give from our digital to analog conversion that voltage we give here, then that voltage will be amplified and you get some current to flow through this load.

So, what is the problem with this interface? So, if you think about what happens we need to go back to the transistor characteristics, the transistor characteristics tells us that unless some bias voltage is there, you cannot fire the current through the collector. So, that means like no you need to provide some bias voltage to the transistor before we expect some current flow to happen in the motor which is one drawback that you have this dead band that will come.

(Refer Slide Time: 27:04)



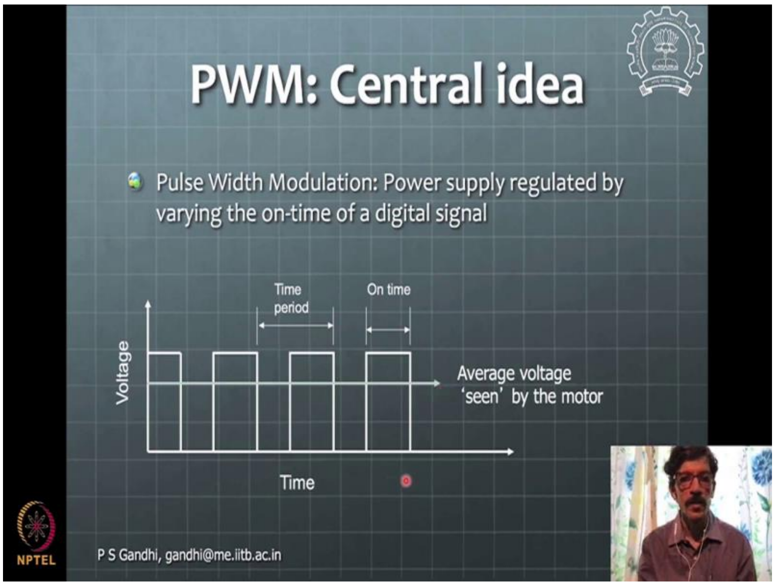
And the other thing that is main reason you will have a problem here that you go below certain voltage given by the digital to analog converter from your microcontroller and then beyond this only the motor will start running somehow. And, other problem is where like when this is very, very low this voltage or the signal that is that is given from the microcontroller is very low, you will have a lot of noise that will be seen the signal to noise ratio will be very high for the low signal low value of the voltage that you are providing.

And that would drive some, your actuators will get driven with that low signal that is the amplified version of that signal will go to the motor and then motor will start making like a noisy behavior. So, this is main problem here and other thing of course we need to think about is the other side we do not have like same characteristics and then that side we will need to see how do we reverse the direction of the motor.

We cannot drive current negative by the same circuit we need to have something else to reverse the direction in the motor. So, how are the solutions for this problem done by using pulse width modulation is what we need to think about and see. So, you pause and think how this problem can be solved by using pulse width modulation where pulse is of a fixed voltage value, but the pulse duration can be changed.

So, think about that and then pause here for a while after you have gathered your thoughts, then you can go forward. So, ponder over these questions.

(Refer Slide Time: 29:02)

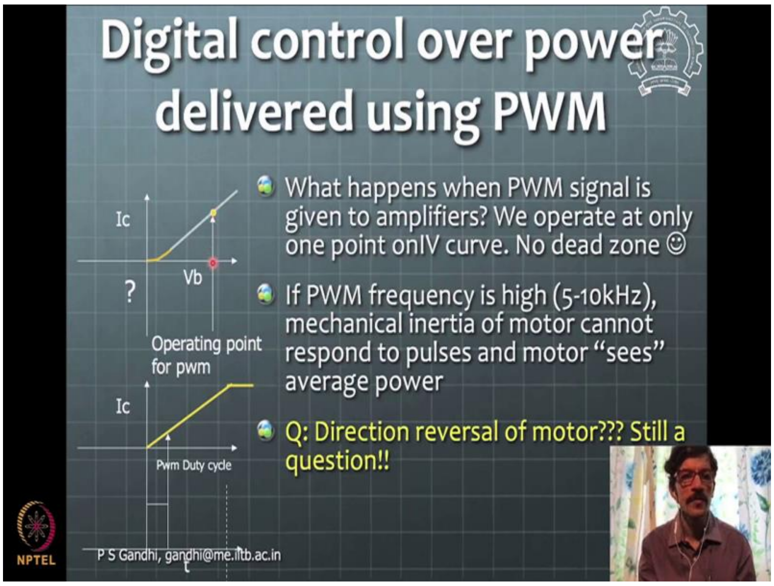


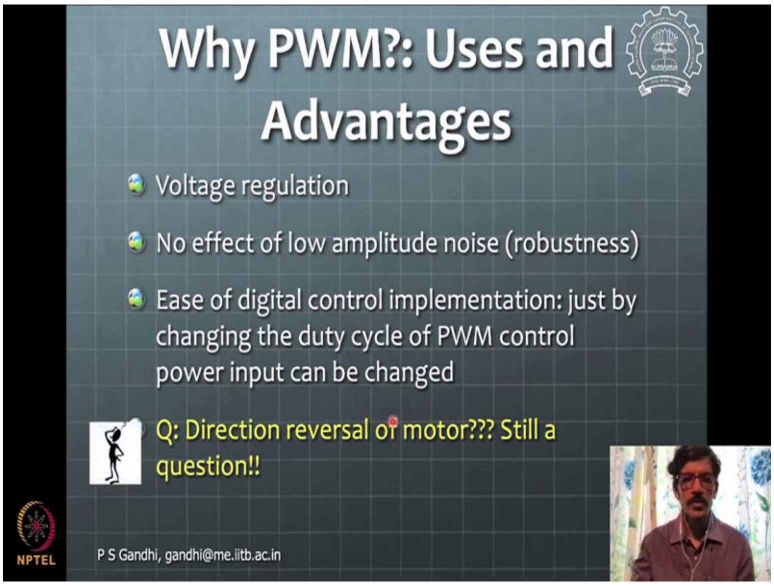
So, as I have told you, the pulse width modulation has this idea where the on-time period can be changed. So, now what happens when we do this thing going to the, to an amplifier the same kind of amplifier we use. So, we use same transistor amplifier, but now instead of giving analog as the input to that amplifier, we give this changing pulse as an input to the amplifier and this pulse is given at a very large frequency.

So, large frequency is that in such a way that your motor output it does not respond to these pulses. What I mean here is that, see these pulses can be changing at a frequency of say a few kilohertz. Now imagine, if your motor is given the input at that higher frequency, motor has a large inertia. Their frequencies of the motor or the environment of the motors will be typically in some few hertz range and when it is given kilo hertz kind of a signal what motor sees actually is average value that is seen here.

So, how this average value is we will also be clear this is based on basically because the inertia of the motor is much higher.

(Refer Slide Time: 30:47)





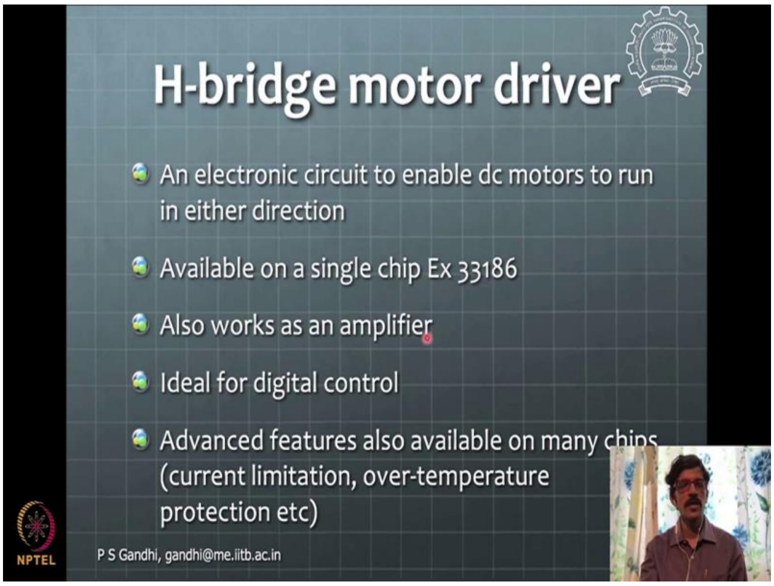
And suppose we give this input to our amplifier, then what happens is the value is always operated, operating point is always here because value is either 1 or 0 you can see that this value here is either 0 or 1. This is 1 here and this is 0 here. But only the period of on time it changed there is a time period and on time out of that is changed. So, since the on time is changed and this value is fixed, now this noise if you talk about that we said is in the small range here.

So, since we are operation operating at this point, the noise is not going to bother us anytime now. So, noise will be much smaller than this value here. So, the voltage value that we are providing since it is either 1 or 0, the noise levels are relatively low as compared to that and that is how we get rid of the noise problem here. And then this, how does, how do we change then, the amount of power given to the motor by changing the on time.

As you give on time lesser and more like you see that the power delivered will be more or less, average value of the current that will be flowing in my motor is going to change based on the on-time or how much time I give this during the cycle of the PWM. So, this mechanical inertia of the motor does not respond to these pulses is what we may make use of this fact and the motor can see then only the average power delivered.

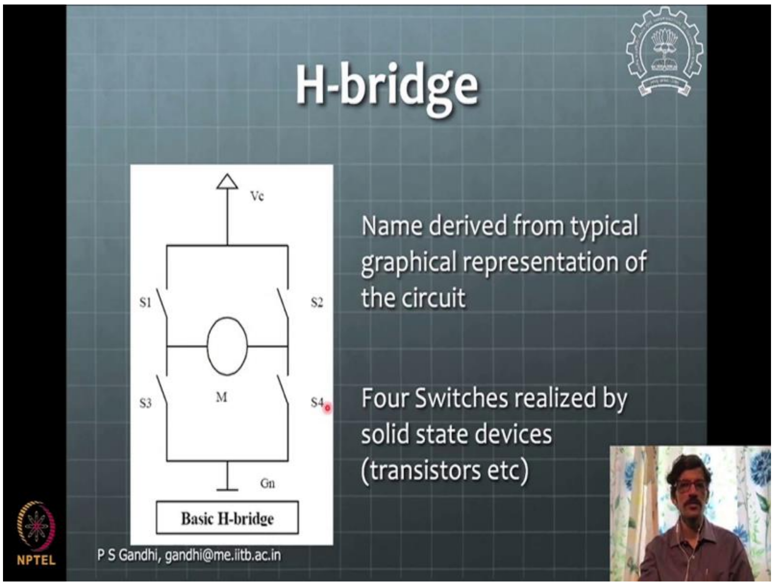
So, this is how this fundamental principle is used for driving all your actuators in a very, very noise free way. And how do we reverse the direction? We use the edge bridge circuitry to do this. So, this you might have already, some of you might have already used these in programming some hobby motors or something like that. So, these are the advantage that are listed for the PWM.

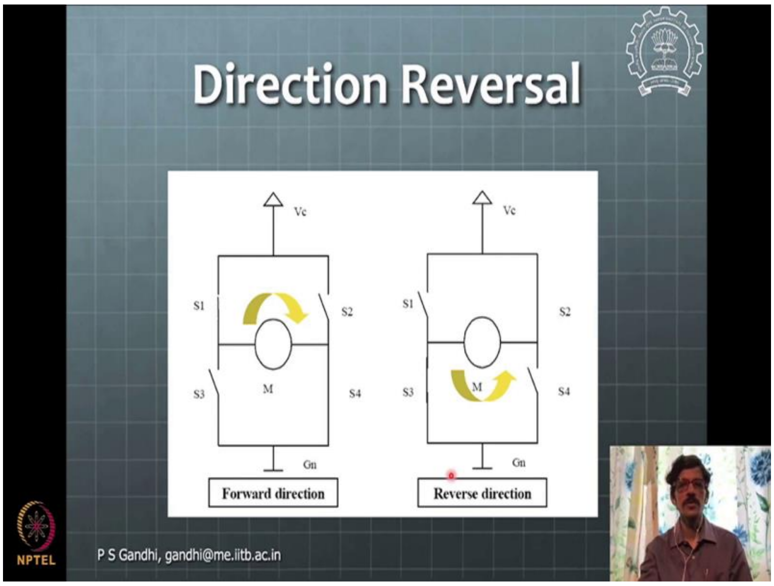
(Refer Slide Time: 33:21)



Then, how do we reverse the detection is using this H-bridge and H-bridge there are these lot of these different, different chips available for H-bridge functionality and it works as an amplifier and additional reversal both functions are done in one chip and this is very good, ideal thing for motor control digital control of the motor. And there are many advanced features could be available in other chips.

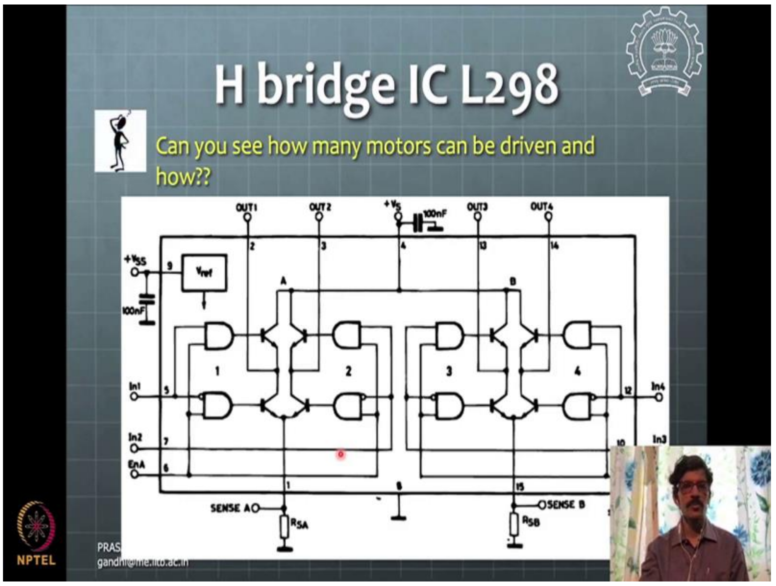
(Refer Slide Time: 33:54)





So, this H-bridge has a very simple way of switching the transistors and reversing the direction of the current in the motor. So, this is transistor-based switches are realized here. And this is one way direction and this is the other way direction reversal by switching S1 and S4 first and S3 and S2 the next or reversing the direction.

(Refer Slide Time: 34:21)

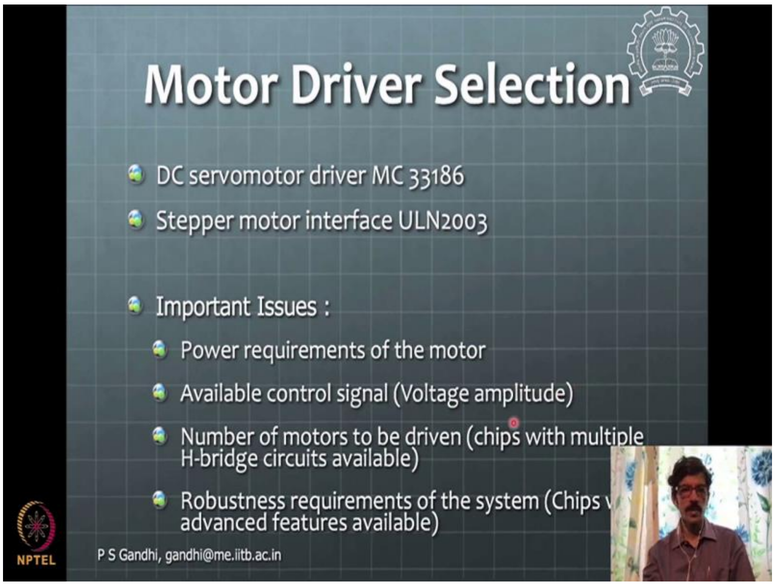


So, you can see this is a configuration of typical H-bridge. So, it has some gates to operate and things like that. And so, one can figure out from this block diagram of that chip, how this chip can be interfaced with your microcontroller and what to be, what is to be connected to motor to drive it. So, think about this, we will have this important exercise to be done.

So, there is inputs here and then there is EnA signal, enable kind of signal here and then there are these outputs that are coming out of these H-bridge. So, you can see how many motors can be driven, see here there are 1, 2, 3, 4 outputs are there. So, you can drive two motors, one motor can be connected here and another motor can be connected here. And then suppose I want to reverse the direction, what should I do? that you can figure out from this diagram.

So, think about this and there are these some sensing terminals also are available here. So, which can be used by user for something. So, you think and figure out it is not very difficult when this diagram is given to figure out how this current is going to flow for one particular direction, which transistor should be on. So, these transistors here you can think of them as a switches and then the switches are operated when these output of this AND gate becomes 1 that is a hint to think about. So, then now you can figure out how things would be working here.

(Refer Slide Time: 36:16)

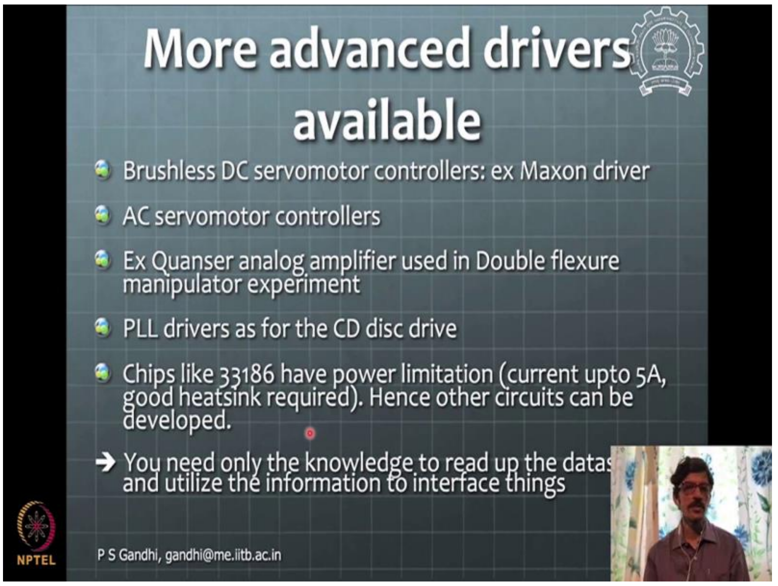


Then, there are many different drivers that can be possible for the, for this H-bridge or depending upon motor. So, permanent magnet DC motor can be driven by the H-bridge or a stepper motor can be driven by this some simple ULN transistor amplifier.

And important issues to consider is power requirements for the motor, available control signal, especially the voltage output amplitude and this power requirement and motor voltage compatibility should be seen, power (not in power), but voltage and current compatibility to be seen. So, driver also should be compatible with the motor. Say if motor is requiring two amperes of current, driver should be providing more than that, I mean typically 3 or 4 amperes of current rating should be there for the driver like that.

And of course, voltage compatibility should be there. Then, number of motors to be driven would be other issue that we can see. Then, robustness requirements. So, there are some advanced chips that are available to or advanced drivers that will be available for giving you some additional functionality.

(Refer Slide Time: 37:37)

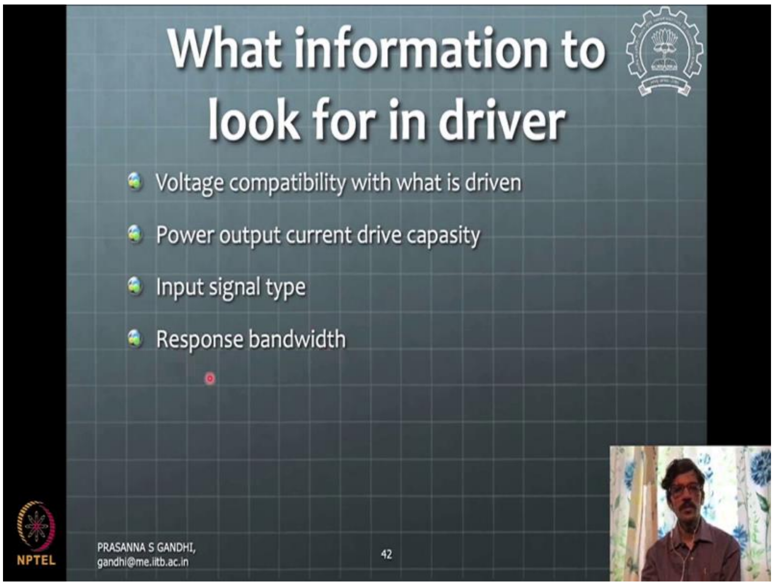


Then typically, brushless DC motors, there are some controllers available for example from Maxon motors or FAULHABER there are two major players in the international market which are having excellent quality for these motors. Then there are many different companies Siemens has the controllers or, so you can look at yourself there are a lot of motor drivers available.

Then, you go for AC servo motors when you have very high power or torque requirements. Then, we use Quanser analog amplifier, this is an analog drive this is not a PWM amplifier, this is an analog amplifier which is used in our double flexure manipulator system where we want to have a very high, very low noise in the system and very high precision positioning requirement is there.

So, like that you can have many different kinds of drivers possible. So, what you need basically the fundamental knowledge is about what we have seen, how do you read the datasheet and utilize the information to interface these different-different things. So, you need to see mainly the current and voltage compatibility and of course, the absolute limits of different things so that you do not damage the systems.

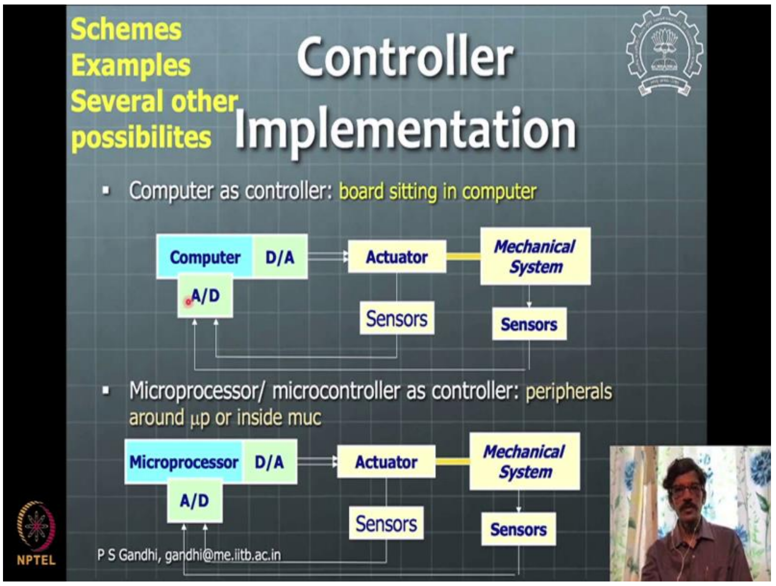
(Refer Slide Time: 39:15)



So, and another thing is very important is it is response bandwidth. So, the driver should be fast enough to respond to, in many places this will not be a problem, especially when we go for MEMEs-based applications, then this response will be a problem because, electronic things respond much faster electrical signals are much faster responding than mechanical motions.

So, respond bandwidth will be satisfied, they will be pretty fast for the electrical domain. But when it comes to very tiny systems like MEMEs, then the motion can be very faster at a low speed, low masses. And then when these motions are very, very fast, then those bandwidths of that motion can be matching with the bandwidth of the controller and that is where you need to bother about the response bandwidth of the controller.

(Refer Slide Time: 40:17)



And these are like some of the schemes for implementation as we have seen these, one of these schemes we have seen already. You can have a computer or microprocessor or microcontroller, whatever you want to use for the brain or for the control of these. So, this is what is about interfacing different-different drivers and actuators here and we will see now how these interfaces can be then further used for building mechatronic application. Thank you.