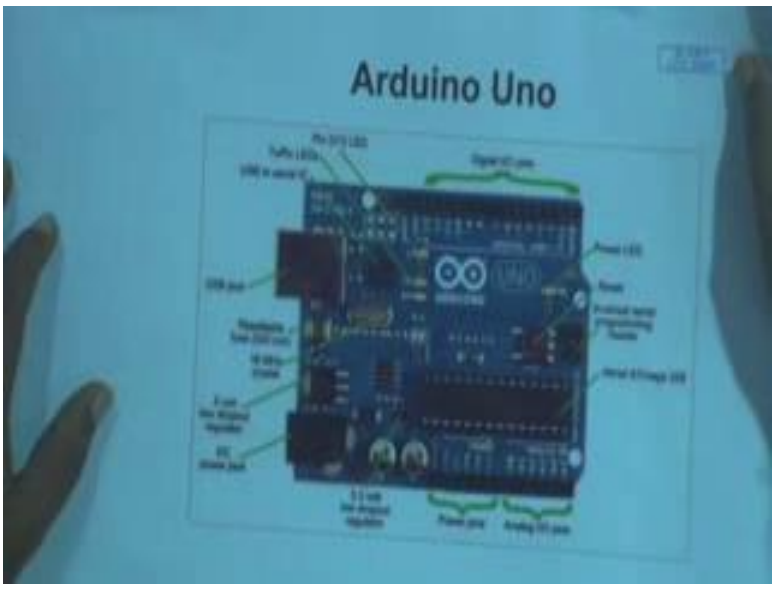
Embedded Systems Design
Prof. Anupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 17
Arduino Uno (Contd.), Serial communication and Timer

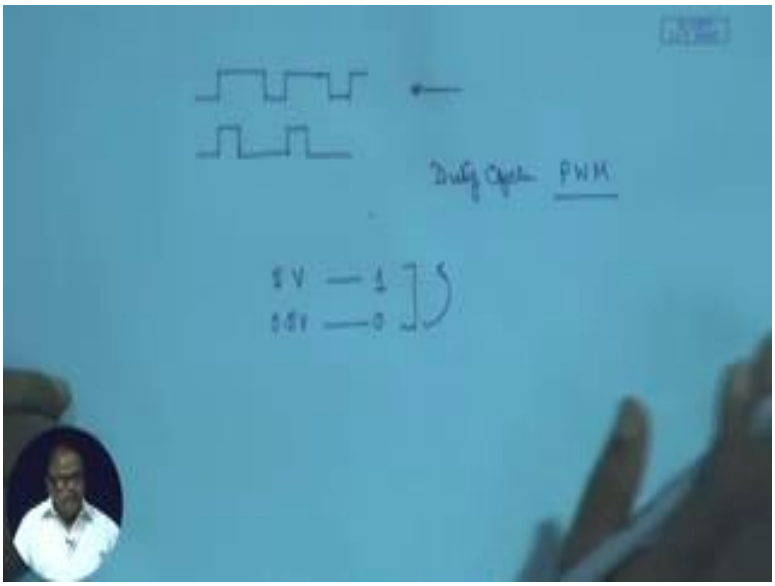
Good morning, we are discussing about the micro controller board Arduino Uno.

(Refer Slide Time: 00:41)



We are discussing about that and we have seen several features of this board Arduino Uno where we have got the analog pins and digital I O pins and we have got analog commands for analog write analog read and most importantly what we saw in the last class is that; it has also got some built in per PWM circuits which can generate pulse with modulated signals.

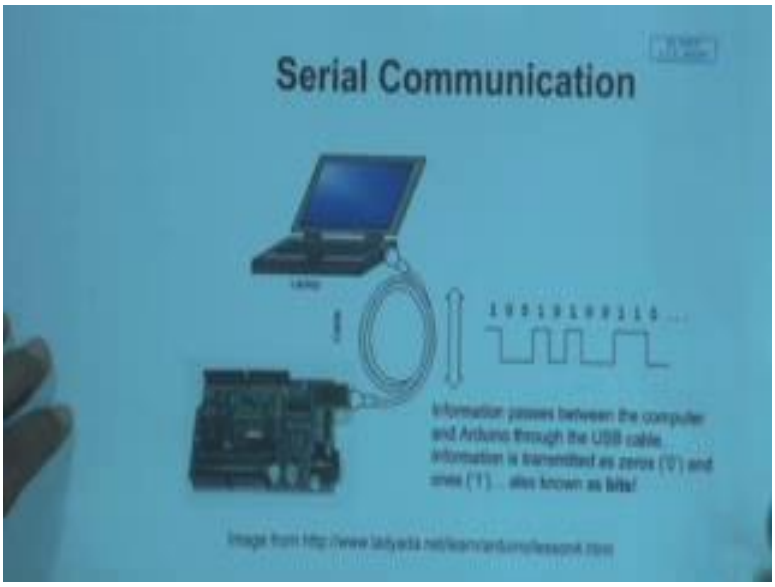
(Refer Slide Time: 01:14)



And by that as we modulate the width of the pulse of say this pattern versus if I have this pattern; obviously, the average voltage is more in this case.

So, accordingly at any output I can change the average voltage by changing the duty cycle of the pulse width modulated system and that is the very nice feature that we have got in our team. Now besides that we will today start discussing about how this Arduino board will communicate with the PC.

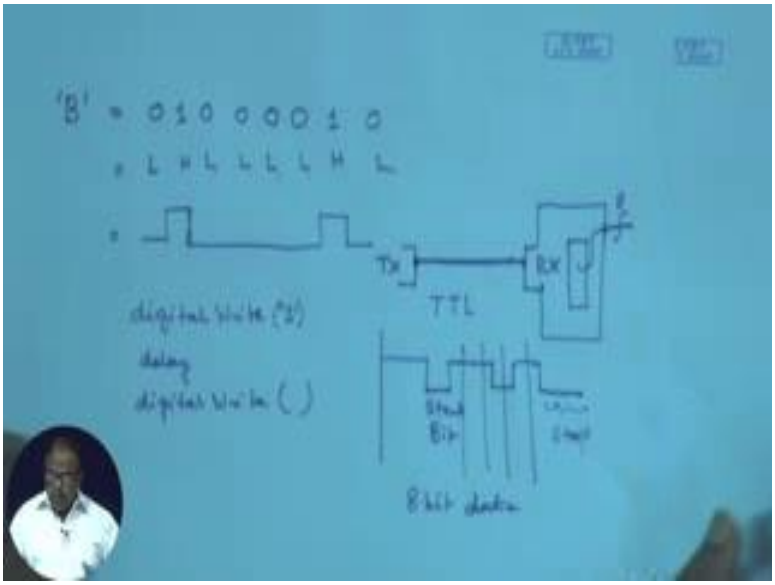
(Refer Slide Time: 02:09)



For example let us look at this scenario; we have got a laptop where even say we see that we will program this Arduino board and for that we will be writing some programs here and that program will be downloaded into this Arduino board and this mode of communication. In this case is serial communication, we will see in detail other modes of communication. Now in serial communication as we can see information passes between the computer and the other host through the USB cable, it can be a USB here it is being shown in the USB, but it can also be in the serial mode. Now USB protocol is a little more complex and for that whenever we are having something serial that serial to USB conversion is required, for that there are separate chips available.

Now, here for example, this serial pattern is 1, 0, 0, 1, 0, 1, 0, 1, 0 etcetera, so this is very familiar to you and usually in the normal scenario we assume 5 volt to view to a 1 and sometimes if 0.5 volt less than with this thing to be a 0 or just the other way around in if it in negative logic that is not.

(Refer Slide Time: 03:59)



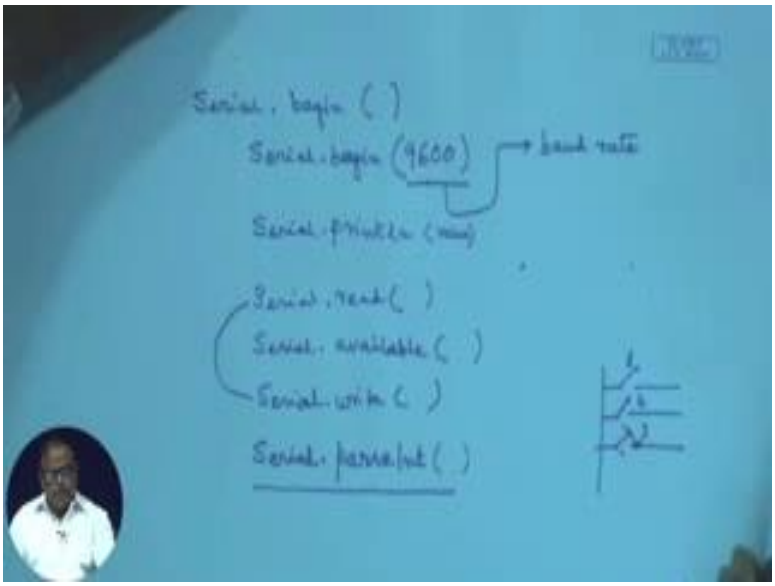
Now, if we look at the say a single character single ASCII code of say B and I am sending that as 0, 1, 0, 0, 0; three 0's; four 0's, 1, 0. So, that will translate through low, high, low, low, low, low, high, low like that. Now, therefore that will be coming through some I will have to send some 0 here, then some 1 here and then 4 pieces of 0 here then one high here and then one low here.

Now when I do this beside, so I am actually toggling the pin to be 1 and 0; 1 and 0 like that, but I can simply do that in this way, that I have got 1 pin and I am just toggling it 1 and 0 or we have got this digital write and where; so how do I toggle it; digital write 1; say 1 and then delay and again I can do digital write, in that way I can do it. Now I need one single where to send the data from this system; to this system. Now this receiver if I call this to be the receiver and this to be the transmitter; the receiver must know when the data has started flowing because this data; say this line is a TTL line; this is a Transistor Transistor Logic being used here.

Now this is a line, when this nothing is floating it is actually in 1; say it is in the high level. Therefore, whenever I want to start transmitting data, I will have to send a start bit first which brings this TTL level to low and after that, it be 1, 1, 0, 1 whatever I go on sending, so that is being sampled at the middle of each of these clocks, so we need a start bits. Similarly since I am sending the serial data, I must also tell the receiver, where is the stop bit; so for that the stop bit is again 2 bits of 0's.

Usually we know that it is an 8 bit data or so I will wait for the start bit and after start bit; I will sample it 8 times and after that I am expecting to stop bits and it goes to stop bits, I can say that particular packet is over. Now very interesting and very simple task that we can do is that our transmitter is sending data in a serial form and in the receiver, where there is a system and there are some buffers etcetera; you convert it into a parallel into a; you are loading it into a register and that is very simple that you can do using a shift register or even in a memory location you can do that. So, then it becomes a parallel data and then if you have some parallel outputs say 8 bits, 8 lines coming out of this then this entire data can be transferred in the parallel form.

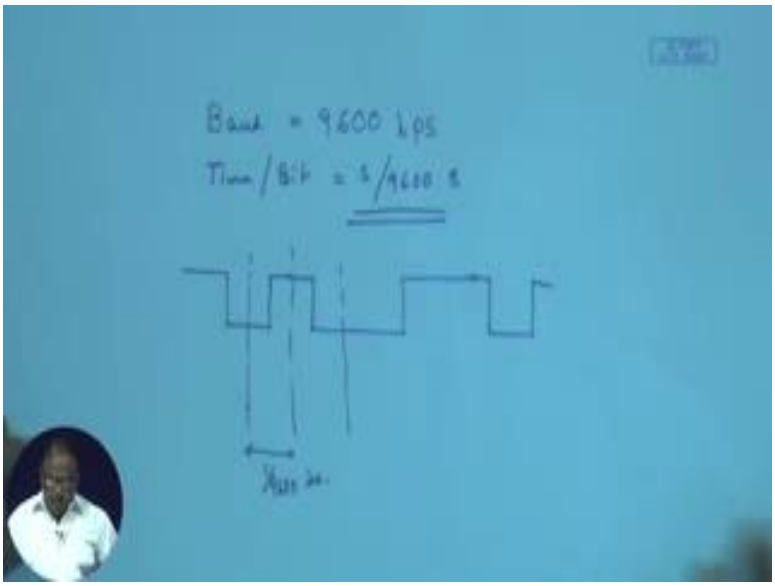
(Refer Slide Time: 08:36)



Now the serial commands you need not memorize it, so for Arduino there are some commands like serial begin for example, we can say serial dot; now what does this mean; this means I am going to send that 9600 bauds; this is the baud rate, that is the big specific, so 9600 bits. So, as soon as I start with this information then this sampling periods are fixed, I will haves so many bits coming per second, so I can very well sense them. Similarly there are other commands like serial print or sometime serial print l n; you can say some value, you can put some value here and you can print it. Similarly you have got serial read, we will use some of these.

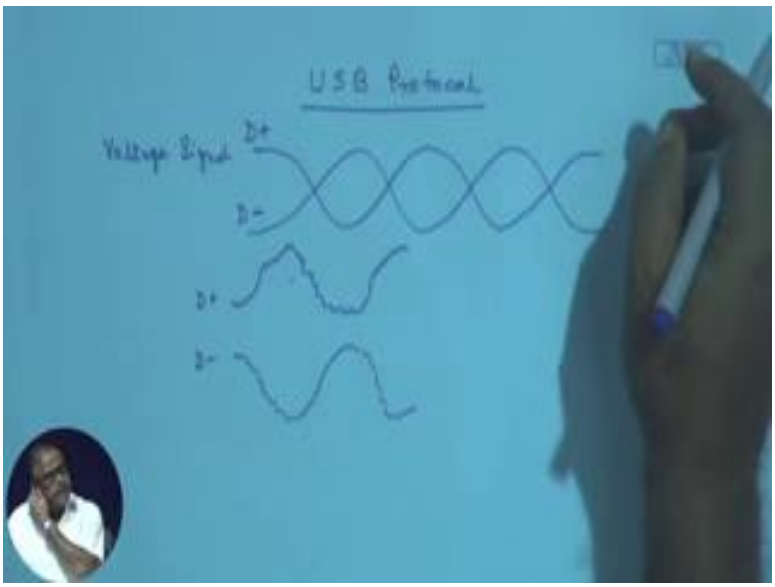
Serial available tells us whether a particular serial port is available, whether it is not being involved in; whether it is being involved in some communication or not. Similarly, with serial read, we will have serial write and one a little uncommon is serial parse Int; this is if there is some event coming from. See for example; a keyboard a particular key there are several keys and each of this keys are encoded, if any of this keys are pressed then the value of this, the value corresponding to this either whether it is key number 1 or key number 2 or key number 3 that will be passed and will be received by the serial port; we will be using it as an exercise a little later.

(Refer Slide Time: 11:38)



Now, we can have two different communication protocols one is of course, serial and we said it is the baud rate is 9600 bps; bits per second. Then we know that time per bit is 1 by 9600 second, so whenever I am having a signal pattern like this; whatever and I am actually sampling at these points at 1 by 9600 seconds, so this time is; right. So, that is how we know whether there is a 0 or a 1.

(Refer Slide Time: 13:06)

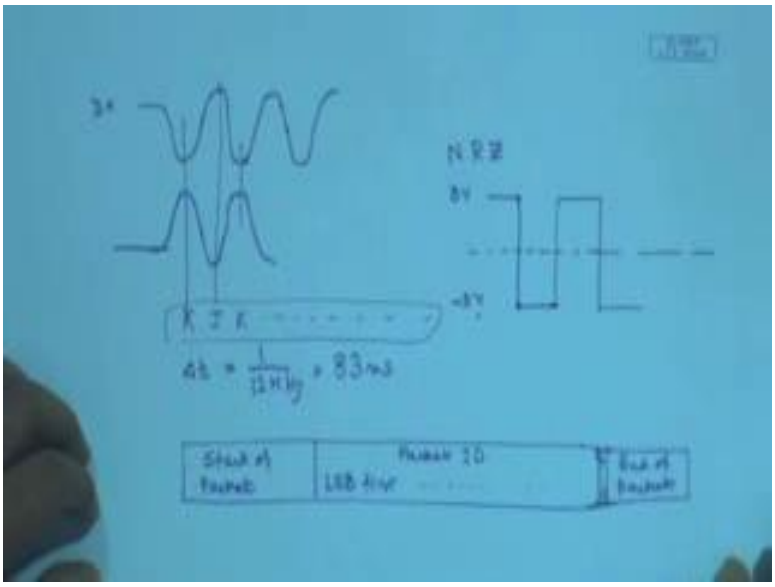


Now the USB protocol one is this is the serial protocol and on the other hand we have got the USB protocol which is other complicated; I am not going into the details of that

here, but one thing that must be mentioned here is the voltage signal in an USB transmission is same through in a differential pair, you know like that is sometimes we use twisted wave as also; let us say I am sending some signal here, some data in a particular phase and here I send the data in the other series; Why? What is the reason for doing that? The reason for doing that is suppose I have got some data.

Let me try to draw some data and that is a D plus analog data I am showing and if there will be another data; I am just sending it in the other phase. Now why transmission? I am transmitting them through some channels and the channels are always having some noise, so the noise will affect both of them. Now if I take the differential of these; I will get the actual data because the noise will be cancelled out because it is being sent in a differential mode.

(Refer Slide Time: 14:50)



So, if I just draw say; some data like this another data is like this; this is D plus another it is coming and is just the opposite of this. Now then from this values, we can see for example, this one is see some K and this one which is you can see the amplitude is more; will send them to as a to d this is might be J some other data alright, this one is again at the same level; this will become K; like that it will go on, so we are doing differential decoding.

Another point is the signals are sent in a non return to 0; non return to 0 means here say I have got some signal and here is my 0 level; this is my 0 level and this is suppose 5 volt.

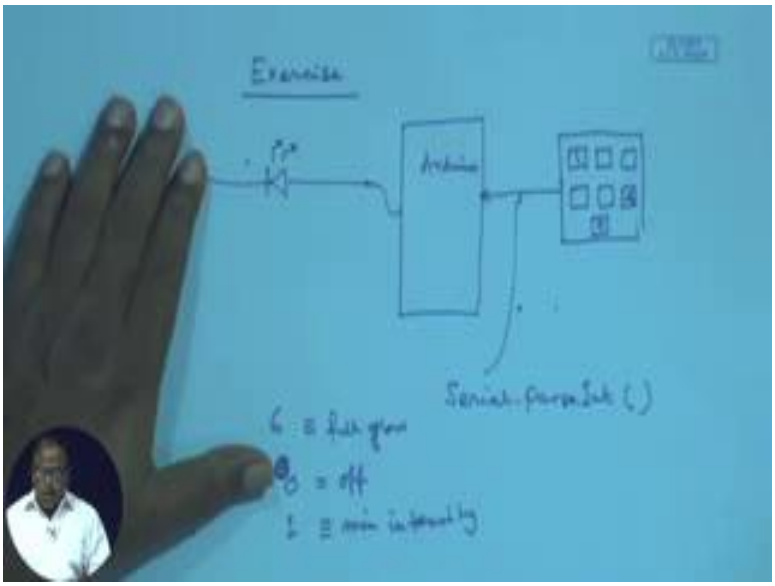
In normal case I had 5 volt and 0 volt or 5 volt or 0.5 volt and here it is minus 5 volt; I am not sending it to 0 alright that has got some extra advantages for error corrections which; we will see later and might be this you already know when you have learnt about (Refer Time: 16:45) coding and all those things.

Now, USB say for example, between each of this data there is depending on the frequency; say I have got 12 megahertz in our case say it will be couplings around 83 nano second is the difference between these two point this thing. Now after this; this entire data that is coming up is boot up in a packet format. Now the packet format is rather complicated, I will just say that there is a start of packet; the start of packet is required for synchronization of the clock; then the packet ID. So, there we send the LSB first; the packet ID will go then the LSB first and then they will end bit and at the end there will be an end of packet.

So, if I have got something coming out from the serial port and I want to communicate through USB then I need a serial to USB converter. Most of you have seen such connectors and what it does is it takes this protocol of the USB protocol and or say for example, I have got the serial bit shown here, I do not know why we give the gap here. So, you have got the serial bit here and that serial bit will have to be encoded with this start packet and Int packet and also we have to transmit it using this non return to 0, non return to 0 and two differential pair and vice versa.

So, but the thing is much more complicated, so we are not going in to details, but you have got convertors available for that, now we had shown I think in the earlier class correct me if I am not that how using the Arduino board, we can blink an LED; we have seen that.

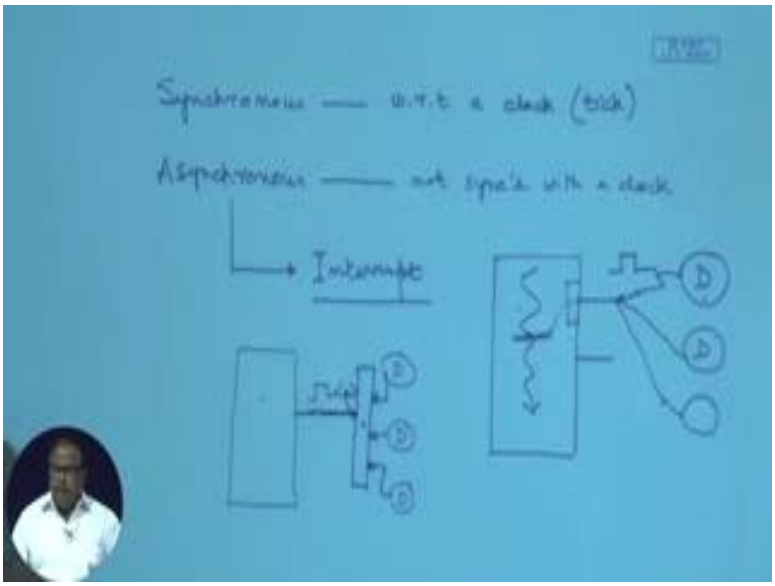
(Refer Slide Time: 19:35)



Now, suppose you want to I am just giving you an exercise to do; now suppose we have got the LED here and I have got the keyboard here and the keyboard has got whatever say 6 piece, so 1 to 6 values of there. So depending on the key that is pressed, we will have the Arduino here depending on the key that is pressed we will glow the LED with different intensities and this thing you can read using the Serial dot Parse Int method and so it comes over here and this LED will glow with different in; I mean when I put 6; 6 means full glow and 0; 0 is not there.

So, let us put a 0 here; 0 is off and 1 is minimum intensity, so you try to do that if you get a excess to an Arduino board or you can try to do that in any micro controller that you get not necessary Arduino is a assignment here, so you can try that. Now whenever we talk of a micro controller, we need to talk about we all the micro controllers communicate one is the serial communication that we know previously, but there are asynchronous communication also.

(Refer Slide Time: 22:02)



What is the difference between synchronous and asynchronous communication? Yes synchronous communication means all the communications are with respect to a clock; a clock tick. So, it may be at every tick it maybe after 5 ticks, every 5 ticks like that, asynchronous means it can happen any time not synched with a clock alright that is asynchronous. Now the best way of one of the very popular ways of dealing with asynchronous communication is interrupt and I think I do not need to repeat interrupts in any way because all of you know what interrupts are; can I assume that? So, an synchronous and asynchronous; now although all of you know what an interrupt is; often I find when I ask the students what is an interrupt. (Refer Time: 23:25) they say interrupt is a pin, interrupt is a signal; whatever it is, interrupt is an asynchronous event that causes the execution of the process to temporarily stop, interrupts are flow of the process.

Now, this can come from a hardware, source it can be come from a software source; accordingly we can have a software can generate also an interrupt from the process or from hardware devices. In the case of hardware devices, we have got the processor and the processor is actually executing the process that is executing here. Usually there are pins through which the external device, some external device sends the interrupt; some signal and that signal is received by this; if the interrupt is not masked deliberately, not deserve will deliberately then that will be acknowledged and this program will be interrupted somewhere and you know that whenever an interrupt occurs, then we will

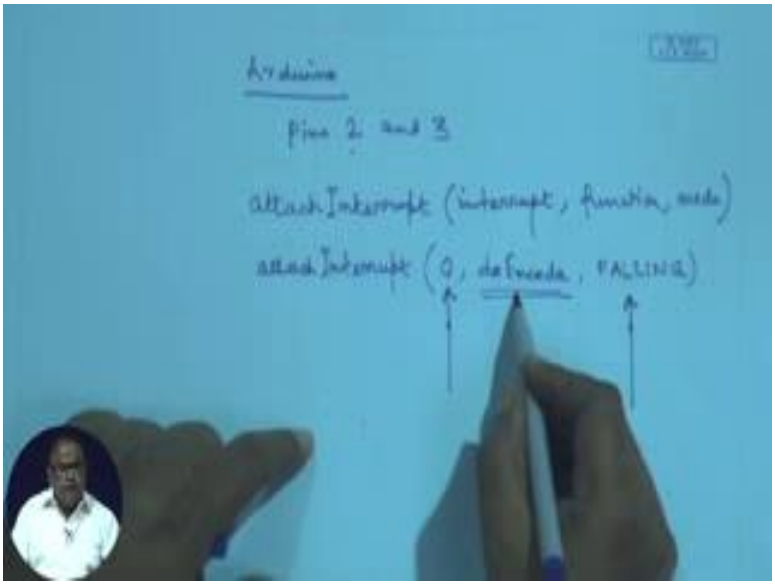
remember the status at this point and then go to some interrupt service routine and again after servicing we will come back.

So what we need sometimes, there is again just since I am talking about interrupts let me complete it. Hardware interrupts can be just like this or they can be a vector, if it be just like it comes to a pin and there to be number of devices which could be connected to the pin; I do not know which devices are interrupted, either this has interrupted or this has interrupted or this as interrupted; I means the processor we will have to check; the flags of each of this who has interrupted.

The other option is to go for a vector interrupt; that means, whenever the interrupt comes; there is a processor and the devices are there and I have got a pin and an interrupt that is arrived at this pin and along with that some more information is sent to me that tells me which devices send the interrupt. So these devices can send the interrupt here and this device; this interrupt controller can connect through the interrupt pin of the processor and feed in this extra information of which devices interrupted; this is known as vector interrupt.

In between we had the solution, if we had individual pins for each of the devices then the thing would be simplified because from the pin itself, there are different interrupts and by the interrupting pin; I could have identified which one is interrupting.

(Refer Slide Time: 26:38)



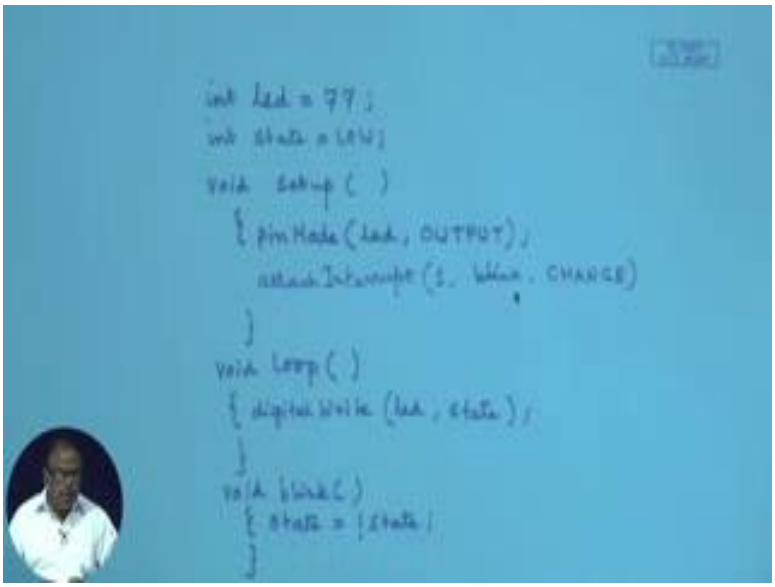
So, interrupt plays a big role in Arduino board; in our Arduino environment pins 2 and 3 can be used as an interrupt pin. We have got special pins like here 2 and 3 can be used as a interrupt pins and for that we have got also Arduino commands attach interrupt from a pin mode.

What is it mean; for example, if I write 0 say encode falling; that means, I am attaching a particular interrupt to the interrupt pin 0 and 1; pin 2 and pin 3 and the interrupt is activated the falling edge; that means, it is an h triggered interrupt, whenever this becomes like this then it is an interrupt and this is the function that I have to execute, this is a interrupt service routine that I will as some interrupt falling a h occurs at this pin; I will interrupts 0th interrupt will be activating the interrupt service routine to encode.

Student: (Refer Time: 28:49)

No; no this is the there are two interrupts, so 0 and 1; so pin is this.

(Refer Slide Time: 29:09)

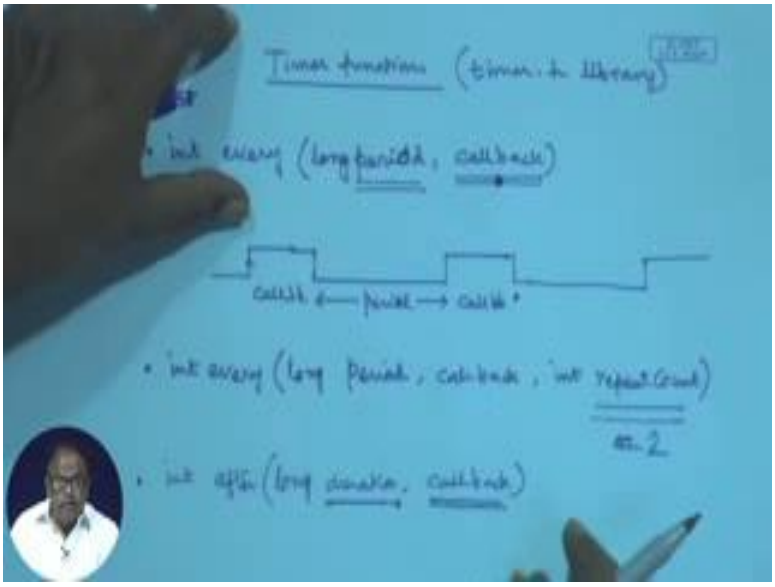


So, here I am just writing a small piece of code for interrupt; I think it will make it clear. State is a variable and I have just set to 0 then setup; remember that we have got the pin mode to set the pins let us say LED with output. So whenever there is a change, I will invoke this interrupt service route in blink, at this interrupt point. So, this is over LED gets state and what was state here? State was low, so and I think I am visible; void blink, it will be state will be the non state.

Now let us see what will happen this blink right, so whenever there is any change at this pin; I will go to this function alright and this is an infinite loop that is going on. So, and I will just toggle the state and then I will come here and digital write that state, so 1, 0, 1, 0 that will happen; at every interrupt. Otherwise if there is no interrupt; this blink will not be activated; therefore, the state the LED will be in the same level. So, this is how we can; this is a very simple interrupt here we have not done any inside this interrupt service route in, we have not done any complicated things, but whenever the interrupt comes; obviously, all the page have I mean fall interrupt handling task set taking care of because this is a very simple way in which we can write interrupts.

Another important component is the timer, so we have got a number of; now this is something that we will come back to our embedded system discussion timer again; time is a very important parameter and we will have to deal with different versions of time.

(Refer Slide Time: 33:04)



So, the timer here there are some very simple commands like timer functions, there is a library just like we include libraries, there is a timer dot h library and we have got commands like Int say function every it is called long period, callback; that means, say callback is a function; that means, the callback is some function that will be carried out every; long period is just like long Int, if after every period. So this function suppose works so that is a regular; this means what? This callback is a periodic function it takes

place here. So, again the callback is started here and then after fixed period whatever that period is; I specify that here, again callback and then there is a same period.

So, this is the periodic systems are very important in real time systems and embedded systems, so I can define a periodic system using this. The next important declaration is Int every long period, callback, repeat count; what is it do? It will carry out same thing, here it was a periodic thing; it will go on and on and on here I give a count say n number of times; we will call this callback. So, this will be callback function this periodic function will be instantiated n number of times.

The other thing is; this was period, but suppose I write after, so in this case what will be the scenario. Suppose a value of n is 2 in that case, this will take run once, run twice and then stop.

Now, if I say Int after long duration, callback what happens is; this call back is activated once after so much duration, so many period milliseconds and returns the idea of the timer event. The other important function is;

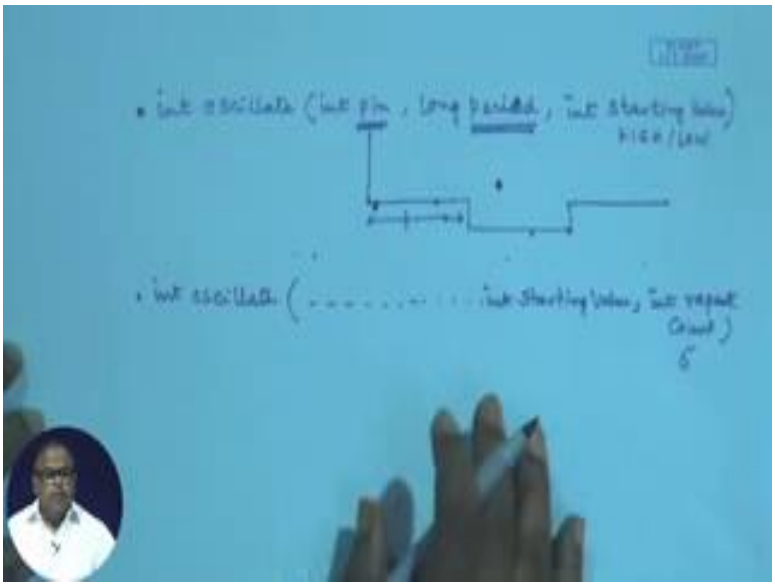
Student: (Refer Time: 37:13).

After a particular duration, the callback function will start.

Student: (Refer Time: 37:23).

Now, if you want to make it periodic in that case we will have to make it; define it as periodic, so whatever is a duration after that it will be activated.

(Refer Slide Time: 37:42)



The other important function is Int is oscillate in pin sorry long period something; that means, pin is a particular pin and what I am saying is after every period; I will toggle the set of the pin. So, suppose this pin was 1 and after this period p; I will toggle it alright and so this p is say in my 1, 2 3 4 (Refer Time: 38:55) and then it will be again toggled up alright. So, it is oscillating and the starting value whether it was start from 1 or it start from 0 is decided by the starting value, which can be this can be high or it can be low.

Now, this also I mean alright; this is used for oscillation, now I can also add this just like this is the same thing Int. Now this means what? It will go on forever, so I can check that also by here saying Int oscillate all the other like Int starting value followed by another integer; that is the repeat count; that means, it tells us that it will go on for so many times it will be so many number of say repeat count is 5. So 5 such oscillations this one oscillation 1, 2, 3 like that, so using this timer function; we can generate wills a different timing events which will come in very handy when we will be handling real time events and real time operating systems (Refer Time: 40:47) and all those things; we will see that this will come in very handy. There are couples of timing functions still left, which we will do tomorrow and after that I will complete the controller.