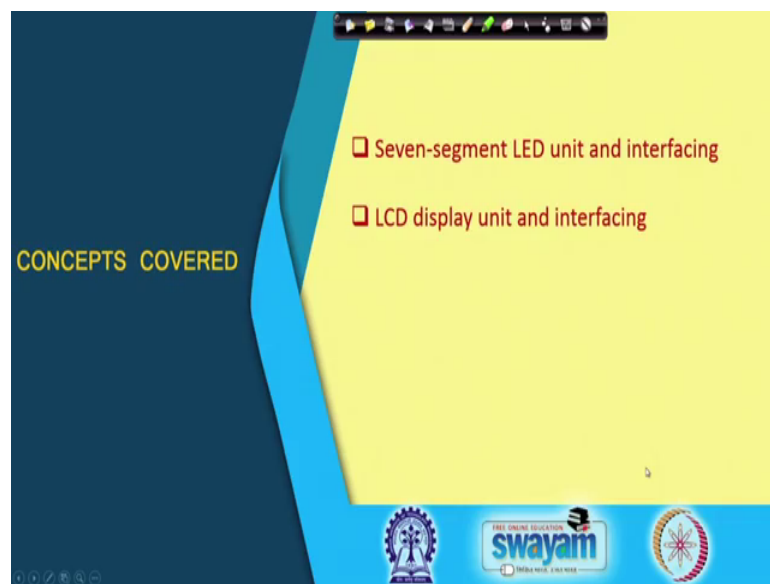


Embedded System Design with ARM
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 14
Analog to Digital Conversion (Part II)

We continue with the discussion on Analog to Digital Conversion. This is the second part of the lecture analog to digital converter part 2.

(Refer Slide Time: 00:27)



So, in this lecture we shall be first talking about another kind of ad converter, which is similar to counter type or tracking type ad converter that we discuss in the last lecture, but is much more efficient in terms of the conversion time. And we shall be talking about what kind of ad conversion facilities are there in the ARM development board that we shall be using, that we have seen that STM32 volt.

(Refer Slide Time: 01:03)

(d) Successive Approximation Type ADC

- The principle of operation is analogous to binary search.
- Suppose we are searching for a number (here V_{in}) in a sorted list.
- We look into the middle of the list – effectively the size of the list reduces by half.
- For 2^n steps, the number of steps required is only n .

128 ✓
64 ✓
32 ✓
16 ✓
8 ✓
4 ✓
2 ✓
1 ✓

$\lceil \log_2 128 \rceil$

sorted

The slide features a diagram of a sorted array with arrows indicating the search process. At the bottom, there are logos for 'swayam' and 'THE ONLINE EDUCATION'.

So, this method of AD conversion that we shall be discussing now is called successive approximation type AD converter. Now the first thing you see what I have mentioned here is that the principle of operation is analogous or similar to binary search. For those of you who are familiar with programming have some basic knowledge and understanding of data structure, you may have come across the binary search algorithm.

But let me tell you what binary search is for the sake of those who does not know it. Just assume that I have a list of numbers I have a list of numbers which are stored let us say in an array and this numbers are sorted in either ascending or descending order; they are already sorted.

Now, I want to search for a number, I want to find out whether a number let us say 45 is present or not. Well if the if the array was not sorted, then I would have to search it from the beginning to the end. I have to look at all the elements, but because it is sorted I can adapt a very intelligent strategy what kind of strategy. Well I can start with the middle of the array I compare whether the middle element is less than or greater than the element I want to search. If I find the middle element is less than that which means my element must be on the right hand side or if it is other way around, then it must be on the left hand side.

So, you see in one comparison I have reduced the size of the list to half. I repeat the process for the half that I have selected. Suppose it is in the right half I again probe in the

middle depending on the probe either I have to see in the left half or the right half suppose I am in the left half I repeat this process.

So, let us say let us take an example suppose I have 128 numbers in the array. So, I start with 128 numbers. Well after one search or one comparison I reduce the list to half it becomes 64. After another search it becomes 32, after another search it becomes 16 like this 8 4 2 and 1. Finally, when there is only one element, then I can tell that whether it is there or not there. You see 1 2 3 4 5 6 7, well 1 you can ignore 1 is the trivial case. So, 7 comparisons are required. So, what is 7? 7 is nothing, but $\log_2 128$ to the base 2. Well we normally talk about the ceiling smallest integer greater than that.

Now, this successive approximation technique is like implementing binary search in hardware in performing the AD conversion.

(Refer Slide Time: 04:33)

(d) Successive Approximation Type ADC

- The principle of operation is analogous to *binary search*.
 - Suppose we are searching for a number (here V_{in}) in a sorted list.
 - We look into the middle of the list – effectively the size of the list reduces by half.
 - For 2^n steps, the number of steps required is only n .
- What modifications are required?
 - We use a successive approximation register (SAR) instead of a counter.
 - The SAR emulates binary search.
 - For example, in 4 bits, it starts with 1000 (i.e. 8).
 - If the input is smaller, the next value is set as 0100 (i.e. 4).
 - If the input is larger, the next value is set at 1100 (i.e. 12).

$\log_2 2^n = n$

The slide features a yellow background with a dark blue curved border on the right. At the bottom, there is a blue banner with logos for 'swayam' and 'Free Online Education'. A small video inset of a man in a white shirt is visible in the bottom right corner.

Let us see how it does. This is what we have already said. Now, you see for 128 I say that it will need $\log_2 128$ right. So, for 2 to the power n if there are 2 to the power n number of elements, I want to search I will be needing $\log_2 2^n$ to the base 2 which is nothing, but how much is that $\log_2 2^n$ to the base 2. This is nothing, but n ; I need n number of steps. So, if I can implement this I need only n clock pulses and not 2^n clock pulses.

(Refer Slide Time: 05:17)

(d) Successive Approximation Type ADC

- The principle of operation is analogous to *binary search*.
 - Suppose we are searching for a number (here V_{in}) in a sorted list.
 - We look into the middle of the list – effectively the size of the list reduces by half.
 - For 2^n steps, the number of steps required is only n .
- What modifications are required?
 - We use a successive approximation register (SAR) instead of a counter.
 - The SAR emulates binary search.
 - For example, in 4 bits, it starts with 1000 (i.e. 8).
 - If the input is smaller, the next value is set as 0100 (i.e. 4).
 - If the input is larger, the next value is set at 1100 (i.e. 12).

Diagram illustrating binary search for 8 in a 4-bit range (0 to 15):

```
graph TD
    A["1000 (8)"] --> B["0100 (4)"]
    A --> C["1100 (12)"]
```

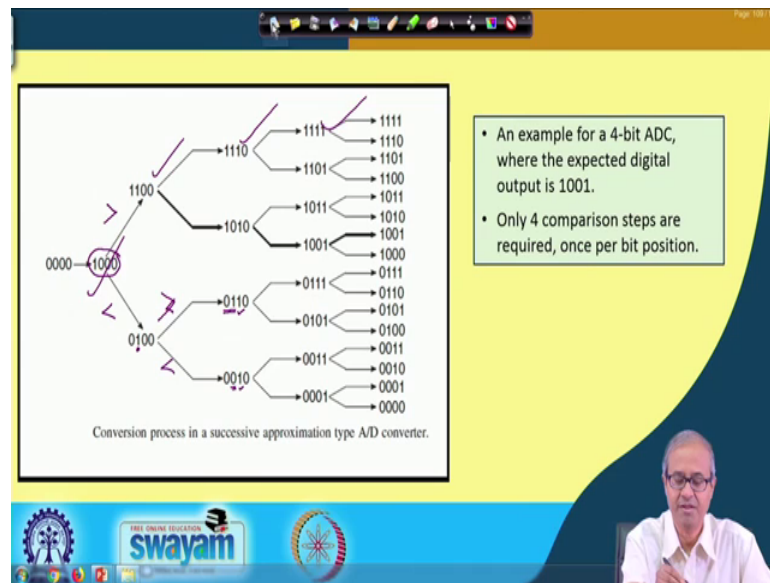
The slide also features the Swamyam logo and a small video inset of a man speaking.

So, how do we do? We use something called a successive approximation register instead of a counter, we do not use a counter we use a special kind of a register called a successive approximation register. What we do? Let us say for 4 bit ad conversion; it is easier to explain.

Let us say we initialize this register with 1 0 0 0 see in 4 bits, we can have from 0 to 15. This 1 0 0 0 means what in decimal 8. So, 8 is approximately the middle of that range 0 to 15; 8 is the middle. So, I start with the middle point. So, I compare whether it is less or greater. So, 1 0 0 0 is 8 right is 8, then it can be less it can be greater. Suppose I say that the input is smaller then it will be on the left half 0 to 7. What is the middle point of 0 to 7? 4. So, what is 4? 4 is 0 1 0 0 and on the right hand side if it is greater, I have to go on the right hand side; that means, 9 to 15 middle point of it is how much 12, 12 is 1 1 0 0.

So, what we are actually doing? We started with one 0 0 0. So, if it is less than this 1 is made 0 and the next bit is made 1, but if it is the other way around this 1 remains 1, I do not change, but the second bit only I am making 1. This is what we are doing repeatedly and this emulates binary search ok. This is the role of the successive approximation register.

(Refer Slide Time: 07:07)



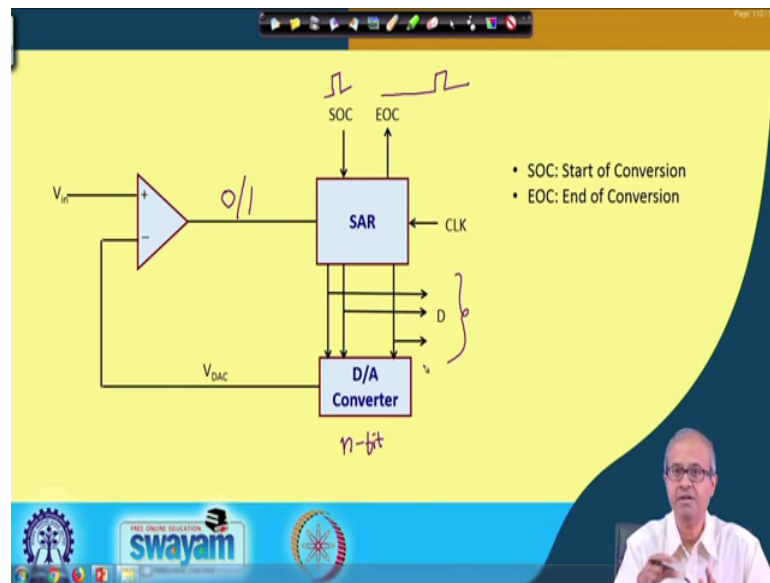
You see this diagram that I am showing on the left hand side; this exactly is doing what I just now told with the example. So, in the successive approximation register we start with 1 0 0 0, we compare with the analog input voltage whether it is less than or greater than.

So, if the if the input is less than that ADC output or the SAR output; then I go here it is less than, it is greater than less than. I make this bit as 0 I make the next bit one 0 1 0 0. If it is other way around, I make it 1 1 0 0 and I repeat this process, how? You see here also I again make a check whether it is less than a sorry less than or greater than. I mean if it is less than in the last bit which I had made 1, I make it 0 and the next bit I set to 1.

But on the other side if it is a greater than this bits, I am not changing the next bit I am changing to 1 and this process I am repeating always I am looking at the next bit either the immediate bit, I am changing it back to 0 and setting the next bit to 1 or I am not disturbing the present bit just changing the next bit to 1.

So, in this way I make 1 comparison, 2 comparison, 3 comparison and 4 comparison and I get finally, my result whatever will be my output of the ad converter right. So, here only 4 comparison steps are required for a 4 bit converter.

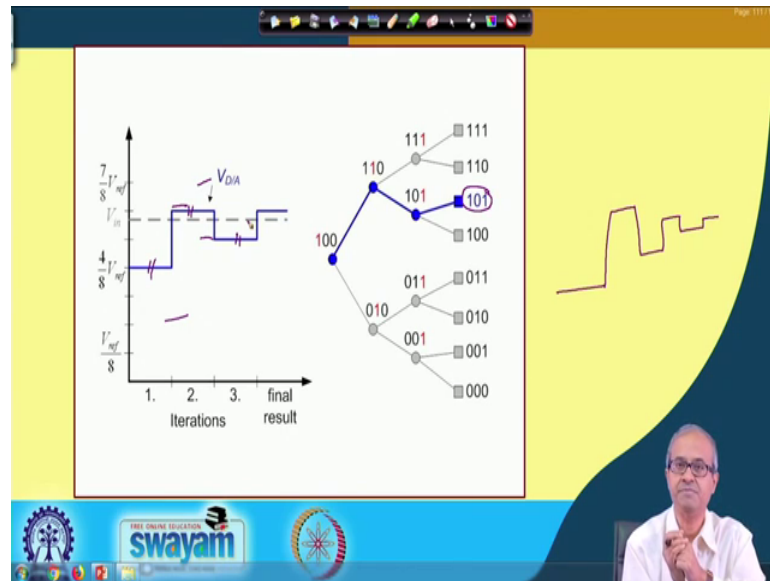
(Refer Slide Time: 08:57)



Block diagram wise this is how a successive approximation AD converter looks like. There is still a comparator in a DA converter very similar to a tracking type ADC just the up down counter is replaced by a successive approximation register and this implements actually the logic, which we mentioned depending on whether the output of the comparator is 0 or 1 it takes a decision.

And whenever you want to start the process of conversion, there are typically two interfaces, start of conversion end of conversion. So, if you apply a pulse in start of conversion the conversion will start and internally there will be 8 clock pulse. Suppose depend suppose you are using an n bit converter. So, you need a DA converter of n bits after n clock pulses; this end of conversion signal will be made high. So, any device outside will know that my AD conversion is over. Now I can read the value of D this is how it works simple in concept, but very flexible and very fast this requires only n number of clock pulses.

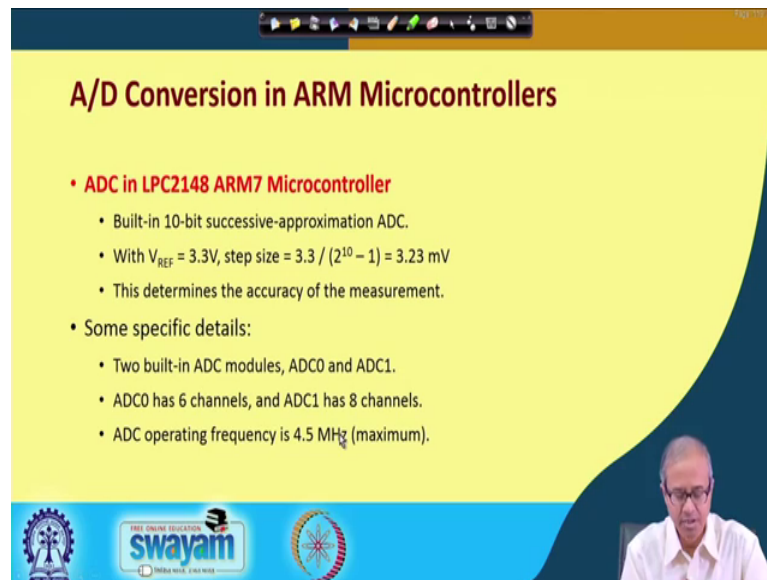
(Refer Slide Time: 10:21)



Just one small example is shown here. This example is for a 3 bit converter. See as this binary search goes on, how the output of the DA converter changes? You see you start with the middle. This is my middle point then you either go here or here let us say I have to go up. So, this is my next point. Next step you either go up here or here let us say you go here like this. You gradually you start with the middle point, then maybe you will be going here, then we will be going here then we will be going like this; you will be converging to the point.

This is a small example that I have shown here for let us say the input corresponds to this 1 0 1. So, it gets converged to that point, but if you look at the wave output waveform of the DA converter output so, what is the waveform? The waveform will look like this fine.

(Refer Slide Time: 11:39)



A/D Conversion in ARM Microcontrollers

- **ADC in LPC2148 ARM7 Microcontroller**
 - Built-in 10-bit successive-approximation ADC.
 - With $V_{REF} = 3.3V$, step size = $3.3 / (2^{10} - 1) = 3.23 \text{ mV}$
 - This determines the accuracy of the measurement.
- Some specific details:
 - Two built-in ADC modules, ADC0 and ADC1.
 - ADC0 has 6 channels, and ADC1 has 8 channels.
 - ADC operating frequency is 4.5 MHz (maximum).

The slide also features logos for Swamyam and other educational institutions at the bottom.

Now, talking about the arm microcontrollers, what kind of ad conversion facilities are there in the arm microcontroller? You see DA conversion is easy, you only need some resistances to make a DA converter, but ad conversion is more difficult. So, you need to have some special circuitry to do AD converter.

Let us look at some of the older generations of ARM. First the ARM 7 was one of the previous generation ARM processors microcontrollers. There was a built in 10 bit successive approximation AD converter. They are all implemented using successive approximation technique because it is efficient, because it requires less amount of hardware.

Now, since it is a 10 bit converter just an example, if I connect a 3.3 volt supply to the difference voltage, then this step size will be full scale voltage divided by 2 to the power 10 minus 1. If you make a calculation this comes to 3.23 milli volt. This actual determines to what level of accuracy you can make the measurement. This resolution is a measure of accuracy with an accuracy of 3.23 millivolt, I can make the measurement when I am reading some analog input and doing some processing, fine.

Now, just inside this processor some specific detail is that there were two such ADC modules, they are called ADC 0 and ADC 1. And this ADC 0 has 6 channels ADC 1 had 8 channels, 6 channel means you could have connected 6 inputs for conversion 8 channel

means you could have connected 8 inputs and the frequency of the clock was maximum 4.5 megahertz. This was the maximum clock frequency that are supported.

(Refer Slide Time: 13:57)

- There is a Control Register that must be written into to determine the operating mode.
- Interrupt is generated after A/D conversion is complete.
- An analog multiplexer is used to select one of the input analog channels at the input of an ADC.

6 { Analog Inputs → Select → Analog Mux → A/D Converter → Digital Output

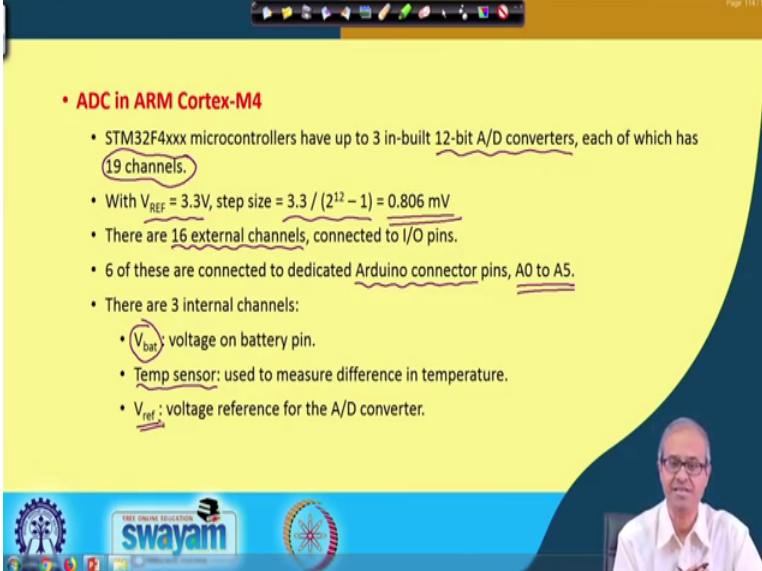
Now, let us see how this channels work. This channels work like this, you see I am first explaining this. Here you have an ad converter and this block that I am showing this is an analog multiplexer. So, you know what is the digital multiplexer is if there are multiple inputs so, one of the inputs are selected based on the select lines. So, analog multiplexer is similar the difference is that the inputs are analog voltages; one of the input will be selected at the output depending on what you are applying at the digital select input lines.

Now, this ADC 0 had said they had 6 number of analog input channels. So, by appropriately selecting it, you could have selected one of them for conversion. Multiple of them you can use simultaneously, but you will have to switch from one to other at a very fast rate; you convert one channel then converts second channel, then convert third channel. Again come back to the first.

Now, this you can possibly use again because of that Nyquist theorem I talked about. Well your AD converter may be working at 1 megahertz speed, but you may not be requiring that kind of a sampling rate depending on your input waveform characteristic may be your sampling rate will be 1 kilohertz only. So, multiple channels you can multiplex on the same AD converter and use them simultaneously. So, here there are 2 such channels ADC 0 and 1 supporting 6 and 8 channels respectively right.

And after AD conversion was completed and interrupt signal was generated ok. This is how it worked and there was a control register which has to be initialized to program that what you want, how you want all details have to be mentioned. This was what was there in ARM 7.

(Refer Slide Time: 16:07)



• ADC in ARM Cortex-M4

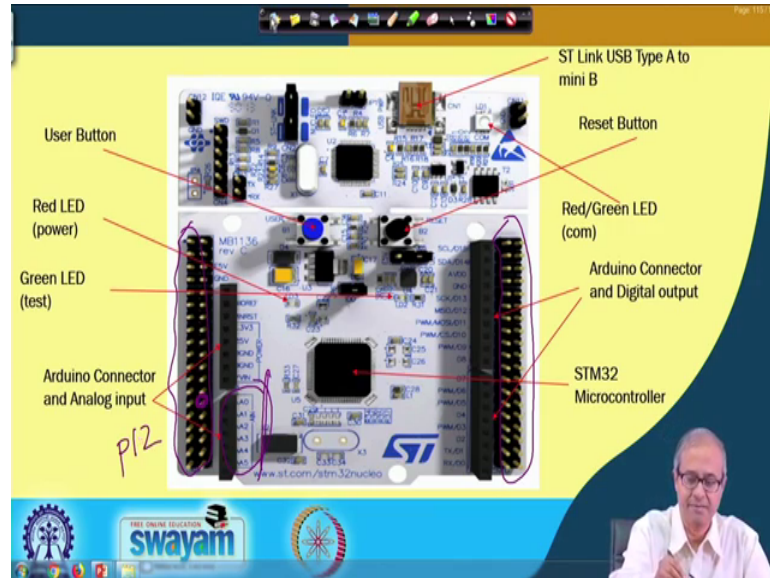
- STM32F4xxx microcontrollers have up to 3 in-built 12-bit A/D converters, each of which has 19 channels.
- With $V_{REF} = 3.3V$, step size = $3.3 / (2^{12} - 1) = 0.806 \text{ mV}$
- There are 16 external channels, connected to I/O pins.
- 6 of these are connected to dedicated Arduino connector pins, A0 to A5.
- There are 3 internal channels:
 - V_{bat} : voltage on battery pin.
 - Temp sensor: used to measure difference in temperature.
 - V_{ref} : voltage reference for the A/D converter.

Now, coming to the board, that we would be using that STM32 which is based on the ARM Cortex -M4 microcontroller. Let us say what is there inside that. Now in this board there are 3 built in 12 bit AD converters and each of them can support up to 19 channels; that means, the analog multiplexer has so many inputs right. So, for 12 bit if your reference is 3.3 volt as you can see, your step size will be much more smaller. So, your accuracy will be much more higher 0.8 milli volts that will be your accuracy.

There are 16 external channels which are connected to the input output pins and out of the 16 channels 6 of them are available on the Arduino connector pins I shall be showing them these are called a 0 to a 5. And in addition there are 3 internal channels which are meant for checking the health of the system the system can check itself. These 3 internal channels are connected to the voltage of the battery if the board is running on battery you can also read what is the battery voltage, then temperature sensor there is a built in temperature sensor in some of the boards. You can read the temperature of the board also and also you can read, what is the current reference voltage. So, all these things are

internal things you can also monitor and read those values; these are called internal channels.

(Refer Slide Time: 17:59)



Now, coming back to this board again, you see in this Arduino connector look at this part. Here A 0 A 1 A 2 A 3 A 4 A 5, the six analog input lines are available here, but more number of analog input pins are supported if you want more you will have to access them from these extension pins. So, in this extension pins STM32 extension pins, you have access to all the signal pins, but here you have access to only the arduino compatible connector pins.

So, when you are requiring AD converter well let us say certainly in the experiments that we shall be showing, we shall not be using too many channels. We can directly connected to the arduino input connectors also. In that case those pin numbers we can refer by A 0 A 1 A 2 A 3 A 4, we can also give those names or if you can use this pin number let us say this particular pin this particular pin number; this will be 1 2 3 4 5 6 7 8 9 10 11 12. We will be calling them as P 12.

And so, you can also refer to that pin as P 12. So, you can either call them by name which is there in the Arduino 1 or you can also call them by the pin number or also port number. I showed that complex diagram; the pin nomenclature ok. There every pin can be accessed by using different names. So, when you write a program when writing a program, you can use any of those names as you want, fine.

So, with this we come to the end of this discussion on AD conversion. Earlier we had looked at DA conversion. Now in the next lectures, we shall be talking about some of the common sensors and actuators that are very important in embedded system applications and many of them we shall be actually demonstrating through the hands on sessions. So, before using them, it is always good to know what the sensors are, how they work and some basic idea as to how they can be interfaced with the microcontroller. So, these discussions we shall be continuing in our next lectures.

Thank you.