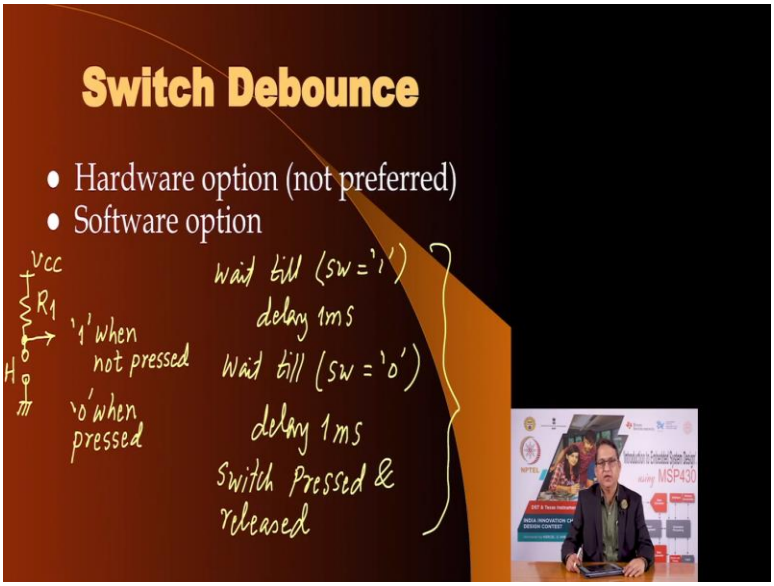
Introduction to Embedded System Design
Professor. Dhananjay V. Gadre
Netaji Subash University of Technology
Professor. Badri Subudhi
Indian Institute of Technology, Jammu
Lecture No. 16
Physical Interfacing - 2

Hello, and welcome back to this session. In the previous session we had seen how we can connect a single switch to a microcontroller pin and just to reiterate and also talk about the debounce activity using software.

(Refer Slide Time: 00:41)



If the switches as you see in this slide, I will read it drawn, if the switch is not pressed, then it will give a logic 1. When the switch is pressed it will give logic 0, but when you press the switch, the switch will bounce and how to handle that in software is what I am going to just write here again.

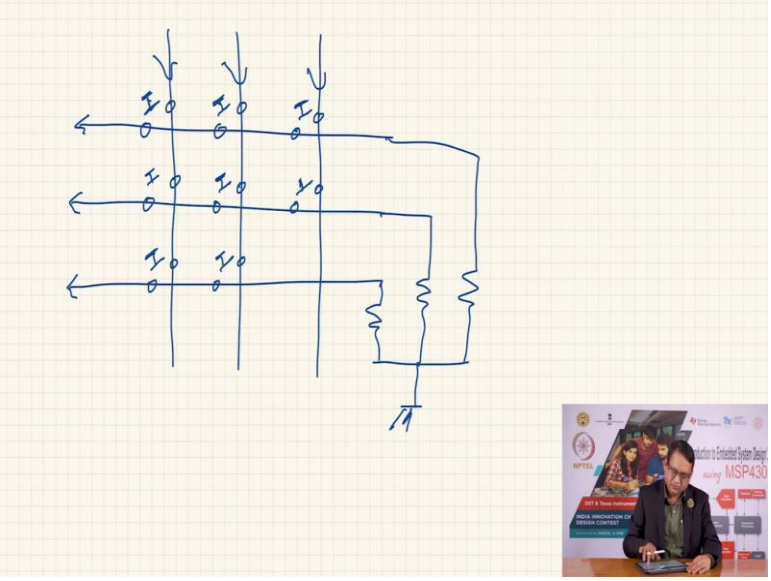
So, you are going to wait till so that we call SW switch is equal to 1. So, if you wait while switch is still the switch is 1, that means you are waiting for the switch to be pressed the moment switches become 0, you are going to get logic 0 you are going to call a delay for we had mentioned say about 1 millisecond then after you come back from the delay the switch will give

you a logic 0 because the duration of time for which you would a human being would presses which would be significantly more than this.

So, you are going to wait for the switch to be released. So, wait till switch is equal to logic 0. And once you find that the switch has been released, you will get a logic 1 again but it will bounce. So, you are going to delay for again 1 millisecond when you come back from the delay hopefully this switch will have logic 1 and now you can go back with the with the understanding switch has been pressed and released, pressed and released and therefore, thereafter you can do whatever you wanted to do in response to this which being pressed.

So, all this will become part of your software as a debounce activity. Now, we have seen how one switch can be handled by the microcontroller. Let us now see if you have multiple switches, and you do not have individual pins to connect to each of these switch. How are you going to handle large number of switches, and that is handled by connecting these switches in a matrix form.

(Refer Slide Time: 02:58)

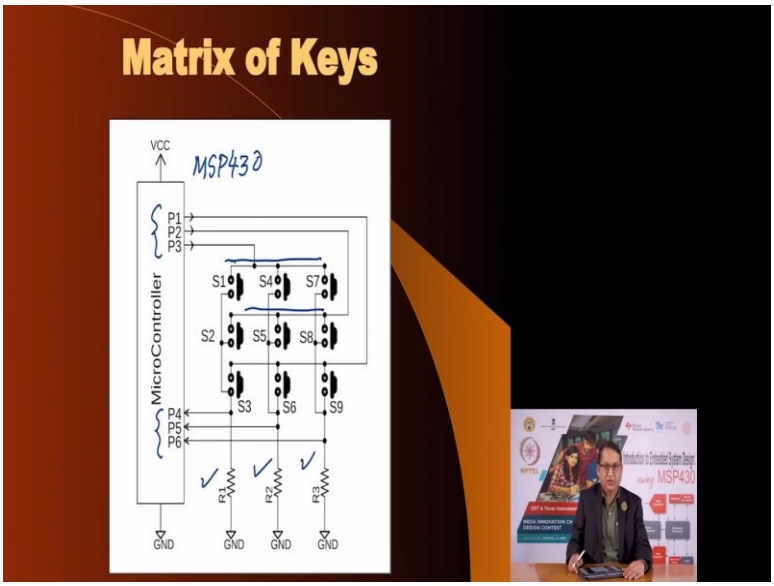


One, the argument behind the matrix is as follows that you would create row and column let us say, I create a row and column of 3 rows and three columns. And I connect 3 registers, which are pull down resistors and these are going to come to columns are going to come from output pins of microcontroller.

And these rows are going to go to 3 pins which will have to be programmed as input and at the junction of each of the row and column you would connect a switch. So, what you have seen is that using 6 pins 3 configured as output and the rest 3 configured as input, you are able to connect how many switches 9.

And this scheme can be extended to cover or connect large number of switches to a microcontroller system. Let me show you a better drawn diagram so that we can discuss this technique further.

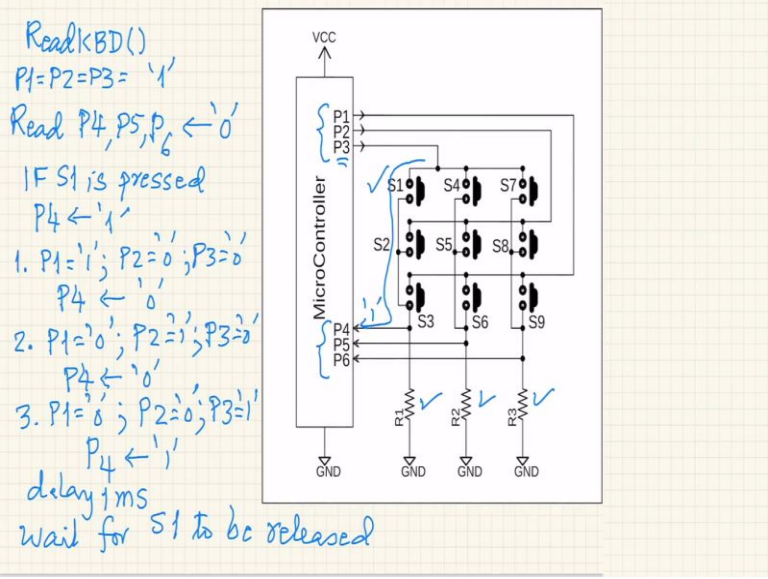
(Refer Slide Time: 04:08)



This is the matrix of Matrix of Keys. Here we see the microcontroller we have is a MSP430 let us say it MSP430 and 3 of these pins P1, P2, P3 will be programmed as output. The other 3 pins here P4, P5 and P6 will be programmed as input. And we see that P3 is connected to these 3 switches S1, S4 and S7 to one end of these 3 switches, P2 is connected to S2, S5 and S8 here and P1 is connected to S3 S6 and S9.

Similarly, P4 is connected to the other end of S1, S2 and S3 and there is also a pull down resistor. P5 is connected to S4, S5 and S6 and there is a pull down resistor R2 and P6 is connected to S7, S8 and S9 with a pull down resistor. Now, you need a bigger software sub routine to handle this activity.

(Refer Slide Time: 05:17)



So, this Read keyboard function let me describe it here with call this subroutine as Read KBD, short for Read keyboard. This function will be called by the microcontroller and it will perform the task of identifying if any switch has been pressed and if a switch has been pressed, what is the code of that switch?

Now, I have brought this diagram here for easy reference. We know that port pins P1 P2 P3 have been declared as output pins. Port pins P4, P5 P6 are input pins and there are 3 pull down resistors R1, R2 and R3. So at the beginning of the subroutine, the subroutine will set P1 equal to P2 equal to P3 to logic 1 and then it is going to read the value of port pins P4, P5 and P6. So, it will perform a read operation for Read P4, Read P4, P5 and P6.

Now in the situation that no switch amongst these 9 switches is pressed, the logic that will be returned on port pins P4, P5 and P6 will be a logic 0. Why? Because we have these pull down resistors. Now imagine that switch S1 is pressed by the user. So, the moment switch S1 is pressed, because the sub routine is reading P4, P5, P6 not only will it read it once it will keep on reading for some time and the moment switch S1 is pressed because P3 is also set to 1.

P1 and P2 are also said to 1. It will return a value on P4 equal to logic 1 all right. So, P4 if S1 is pressed by the user P4 is going to return a logic 1. However, if you notice P4 is not only connected to one end of S3, it is also connected to one end of a S2 as well as S1. Now, it was S1

which was pressed which made P4 equal to logic 1, but the microcontroller has no means of knowing whether it was because of S1 being pressed or S2 being pressed or S3 being pressed.

And so, it must do something to identify that what is the reason for P4 getting a value of logic 1? So, what it will do is it will go through a sequence of operations, the first operation will be 1, it will set P1 equal to 1, P2 equal to 0 and P3 equal to 0. Now, as you see the moment P3 is set to 0, if S1 was pressed, it will make P4 equal to 0 but P1 is 1 but because S3 is not pressed P4 is not going to be 1 P2 is 1, but because S2 is not pressed P4 is not going to be 1, to be 1.

And therefore, it would get P4. So, in this situation P1 equal to 1, P2 equal to 0, P3 is equal to 0, the value of P4 will return a logic 0 again. Then it will go to the next step of identification, it will do P1 equal to 0, it will reset P1 to 0, it will set P2 equal to 1 now, and it will remain retain P3 equal to 0 and then read P4 again and because only S1 was pressed, the value of P4 will still return a logic 0.

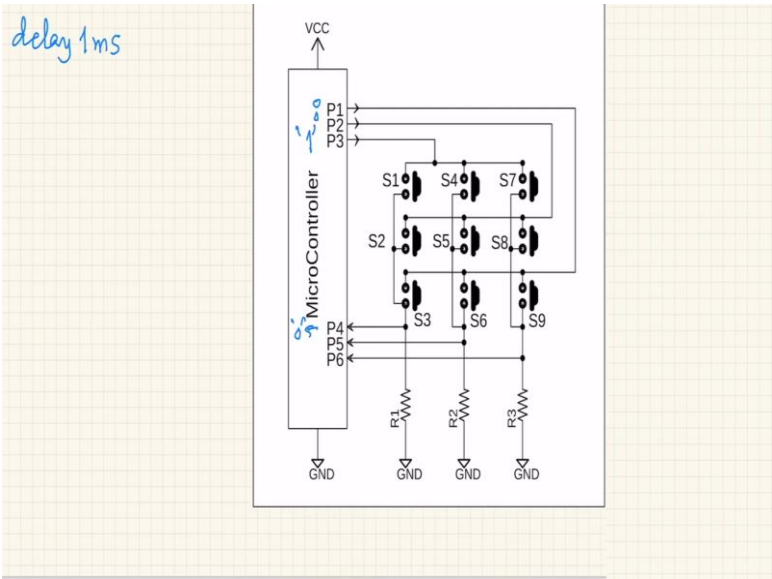
Then it will go to the third step because there are only 3 combinations for P1 P2 P3. It is going to be may retain P1 is equal to 0, it is going to make P2 now equal to 0 and it will now set P3 equal to 1, the moment it sets P3 equal to 1 again, because S1 is pressed the value of S1 will the value of P3 equal to 1 will come to P4 as 1. So, now P4 will get a value of 1, the microcontroller will read that P4 is 1 and from here it will conclude that S1 was pressed.

Now, you see the human operation of pressing a switch versus the microcontrollers operation of performing these tasks, there is a huge difference of time, a human will press a switch and that switch press will last for several milliseconds or several 100 milliseconds whereas the microcontroller can perform these tasks quite fast. And so, even when the switch is just pressed the microcontroller would have concluded that S1 has been pressed.

However, it must wait for this switch to be released. So, what it is going to do is it is going to wait but before that when you press a switch as we have mentioned there is a bouncing on the switch. So it has to debounce that switch so it is going to delay called sub routine or delay sub routine for 1 millisecond. After that, it is going to wait for wait for S1 to be released.

And it may take any amount of time because people would they have different habits somebody may press it momentarily some way somebody may press that switch for a longer duration of time, the microcontroller has to wait till such time that this switch is released.

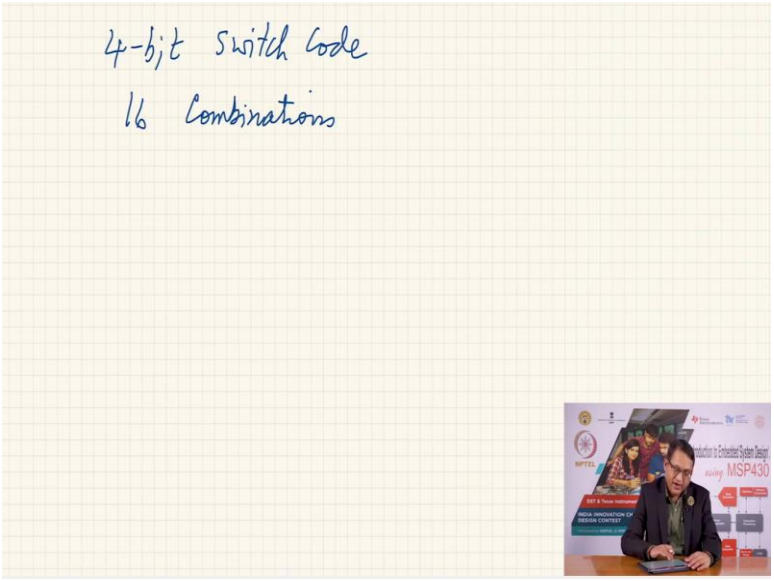
(Refer Slide Time: 12:02)



Then what it will do it is going to again, when the switch is released again the switch debounces, switch bounces and so it must debounce it so it is going to again called a delay sub routine of 1 millisecond.

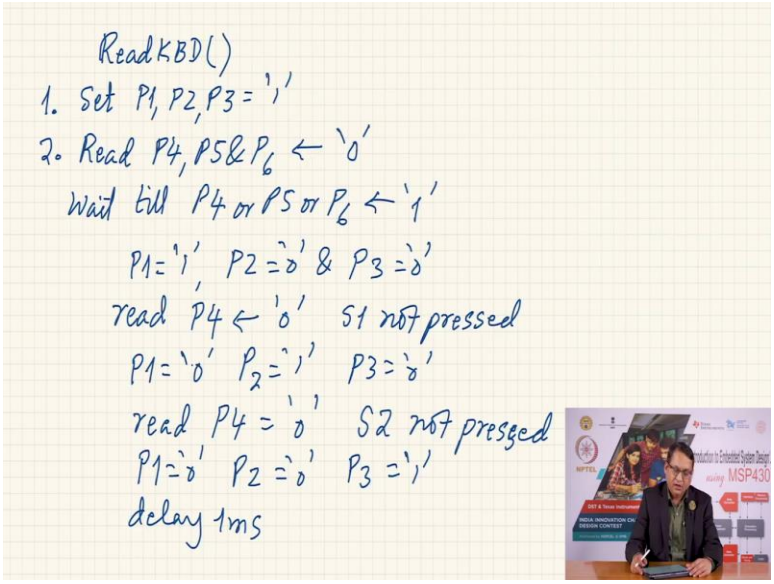
And after that it will check the value of P4, while P3 is still 1, P3 is 1, P2 is 0, P1 is 0. If the switch S1 is released now P4 will return a value of 0. And so now it will conclude that yes, S1 was pressed and it has now been released. Now it has to send the value of a code so that the main program can be informed that it was switch S1 that was pressed.

(Refer Slide Time: 12:52)



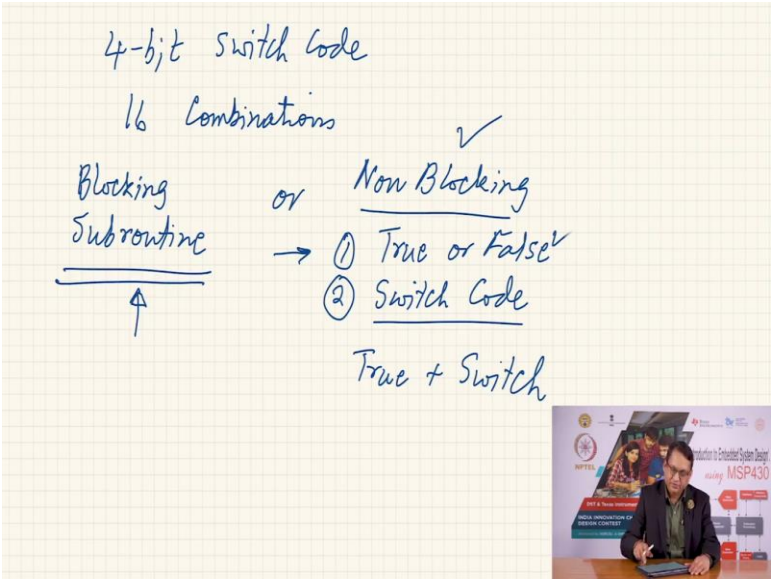
Now how 9 switches using a 4 bit switch code. And one of the values will be for S1 and so on. So, 9 values will be taken using a 4 bit code, you can have 16 combinations, 16 combinations and out of these 9will be used, the rest are not to be utilized and so, the main program will get the information that which of the 16 which has was pressed if any switch was pressed.

(Refer Slide Time: 13:32)



Now, when you create a sub routine like this, this sub routine can be written in two form

(Refer Slide Time: 13:36)



And now I am therefore, talking about the software aspects. Any sub routine such as this which is going to be used by the microcontroller to read external inputs can be written in two forms and they are called Blocking, Subroutine or the same Subroutine can be written as a Non Blocking subroutine. What is the difference?

In a blocking sub routine here, when the subroutine is called the subroutine will not return till a activity is detected, which means if I call read keyboard sub routine if I design it to be of the blocking nature till a user presses a switch, it is not going to return to the main program. What will be the return value of such a blocking subroutine? Only a switch the code number which would indicate which switch was pressed.

But why is it called blocking because it blocks the performance or blocks the operation of the main program. Sometimes this may be a desirable method of writing a subroutine, but in the case of keyboard, this may not be a very good way and instead we program we write the subroutine in a non blocking format.

Now, a non blocking subroutine would have two return values. First value will be one of the values will be the switch code and the let me not say that let me say that the first value will be a True or False and the second value will be Switch Code, switch code, what is the interpretation? So, when the non blocking will wait for default amount of time for the user to press a switch.

If during that time the user does not press a switch, it will return to the main program with the one of these values set to false to indicate that knows which was pressed and in fact, there is a timeout and the value of switch code will be ignored by the main program. On the other hand, if during the default time that is the time for which the subroutine is going to wait for the switches.

If during that time, which we call as a timeout period within the timeout period if a user presses a switch, then after it has detected which switch was pressed, the subroutine will return the true and false value equal to True plus the Switch Code.

So, the main program when it calls a read keyboard sub routine, which is written in a non blocking format, when the subroutine returns, it is going to first check the value of this true or false return value if it is set to true then it will understand it will know that a switch was pressed and it will find out which switch was pressed by looking at the switch code.

And this is the most common method of writing a subroutine for a Switch Matrix. Imagine if the switch matrix such a switch matrix was used in an ATM machine. If it was written in the blocking subroutine method, it is going to block that resource completely. Because suppose you insert your ATM card into the machine and you refuse to enter, enter your passcode the machine is going to wait till you enter it so this is not a very advisable method.

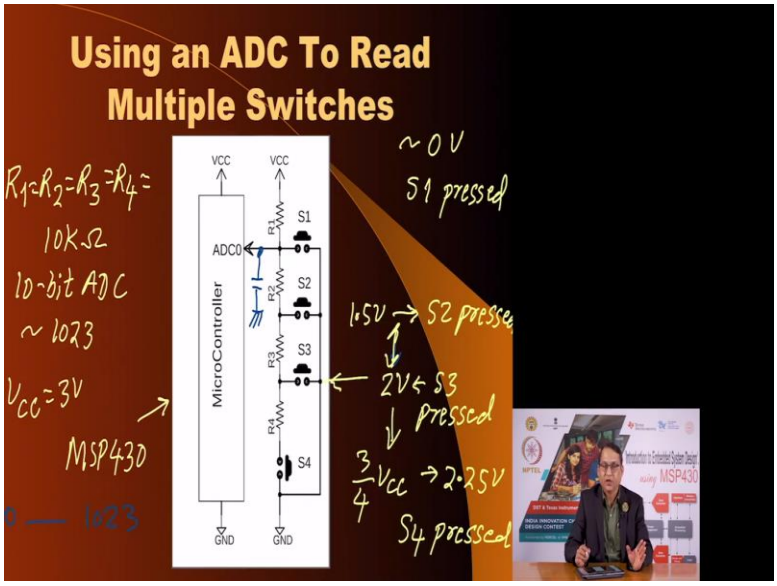
And you would have, you would notice that when you use ATM machine when you insert your ATM card, it asks you to enter your PIN number. If you do not enter your PIN number for a specified period of time, the ATM machine returns your card and says timeout. This is why this is the reason this is to indicate that the subroutine for the keypad on ATM machine was written in the non blocking fashion.

And so we will see in our subsequent lectures, we will show you how to write this non blocking sub routine for a switch matrix if you would like to interface. Let us go to the next aspect. Now, we have seen that using switch matrix you can connect several switches to your microcontroller system. There is one more interesting method of connecting multiple switches and in fact using even lesser number of pins.

In the first configuration, you saw that you could connect one switch per pin. In the second configuration creating a switch matrix you could connect multiple number of pin switches to a

few number of pins. But here is an alternative method which uses a single pin however, that pin should have the capability of reading analog voltages.

(Refer Slide Time: 18:43)



So modern microcontrollers, as we know have if this is MSP430 or and similar microcontrollers if this is MSP430 which as you know has several ADC inputs. using one of the ADC inputs, you can connect as I have shown here, you can connect 4 switches, but there are some additional hardware required as in this case, we need 4 additional resistors and let us say we decide the values of these 4 resistors to be equal values and let us say the values are R1 equal to R2 equal to R3 equal to R4 is equal to 10 kilo ohm each.

Now, how is this system going to work you are going to write a subroutine where you are going to continuously sample the ADC value. Now, when no switch is pressed, what will be the ADC voltage? Will be because of the pull up resistor it will read almost VCC which means the ADC will have a maximum count if it is a 10 bit ADC, which MSP430 has, then you can expect that the count on the ADC will be about 1023 or near about that value or 1024.

Now, if S1 is pressed because the other side of S1 is connected to ground, it will give a 0 volts on the ADC. So, you would know if the ADC input is 0 volts if the ADC input is more than thousand that means no switch is being pressed. If the ADC input is near 0 volts, that is count near 00 a few numbers that means S1 was pressed.

On the other hand, if S2 is pressed the voltage at this junction, the ADC input will be half of VCC because all these registers are equal. So, you can expect VCC being 3 volts, VCC, VCC is 3 volts so you will get about 1.5 volts at the ADC input. So, the correlation count corresponding to 1.5 volts you can expect. So, when you get 1.5 volts, you can conclude that S2 pressed.

If on the other hand S3 is pressed, that means this part of the resistor R3 is grounded, that will give a volts voltage at the ADC input which is 2 by 3 of VCC. That means you are going to get roughly 2 volts that means S3 pressed. Similarly, if I press S4 it is going to put an ADC value equal to 3 by 4 of VCC which is about 2.25 volts. And so if you read the ADC count corresponding to 2.25 volts, then you conclude that S4 is pressed.

And of course, here also you will have to debounce and perhaps you could connect a capacitor from here to this, so that it will filter the ADC input and it may also be used for debouncing the voltages because of this switches being pressed.

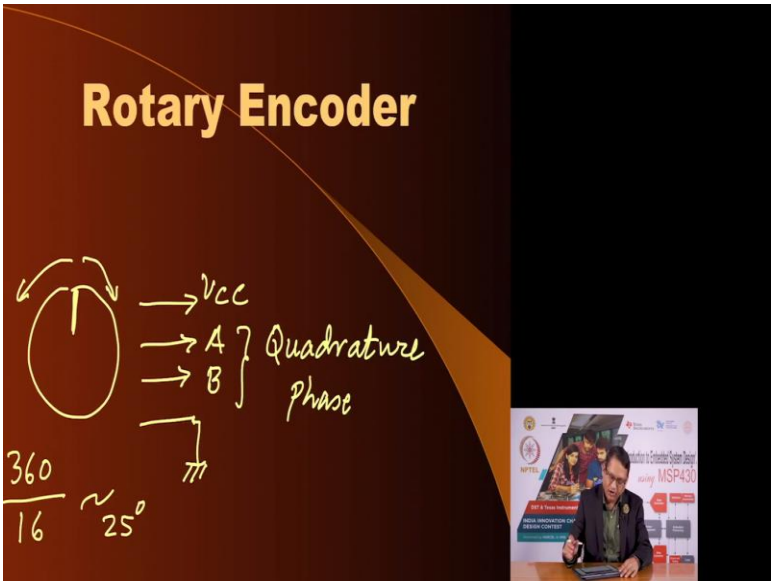
But you can still add a delay and then wait for the ADC values to stabilize to conclude whether S1 is pressed or S2 is pressed or S3 is pressed or S4 is pressed and you can extend this technique, but you should not you may not be able to connect many, many too many switches because the, the entire range of the ADC, in this case 10 volt the 3 volt range has a count from 0 to 10 to 3. As you add more and more switches you would get a smaller range of numbers.

And in fact using equal value resistors of 10 or whatever kilo ohms each is also giving you a nonlinear counts meaning when switch S1 is pressed compared to when nothing is pressed you get an entire variation from 3 volts to 0 volts. Whereas, when S2 is pressed you go to half as you see the difference between the voltages here, this is less, and even less and so on. And so, one method could be to appropriately select the values of R1, R2, R3, R4 to be non equal values.

So that you get a voltage range which is equal parts from VCC to 0. And yet, one does not recommend this technique to connect more than four and five or maybe seven, eight, ten switches at max. So, this is an interesting method to connect multiple switches on a single ADC pin of a microcontroller.

Let us see what all we can do. Now the next input device is a Rotary Encoder, what is a Rotary Encoder? Usually it is a knob and this is how it looks physically,

(Refer Slide Time: 24:17)

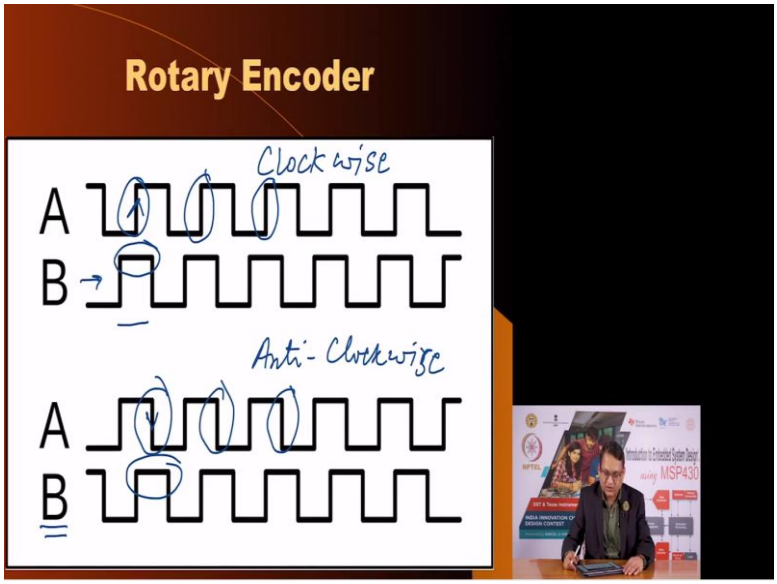


But electrically it has 4 pins. One of the pins is VCC that means you have to connect it to VCC. Another is ground and there are usually two outputs labeled A and B. What it does is let us say this is the reference point of this Rotary Encoder, as you move it either clockwise or anti clockwise, it would give you pulses on A and B outputs. And the rate of the pulses that you are getting will indicate that you are moving the Rotary Encoder fast.

If the rate is low, that means you are moving it slowly for so let us say, suppose this is the 360 degrees that is one entire rotation is suppose this is a 4 bit encoder that means, you have 16 parts, which means for every how much is that roughly, roughly 20 degrees 25 degrees of rotation it will give you 1 pulse.

So, if you move this 25 degrees from the reference it will give you 1 pulse, but you would not know have you moved it in the clockwise direction or anti clockwise direction and for that, that is the reason why these two outputs are provided these two outputs are called Quadrature Phase Outputs. Which means the pulses on output A and B are 90 degrees apart, let us see how these pulses look like.

(Refer Slide Time: 26:06)



So, I have drawn the two sets of pulses, these are the output on A and B in one case this could be, this could be clockwise motion of the encoder may give you this relationship between A and B as you see, if you consider the reference to be B, that means during the time that B is high, A is going from 0 to 1.

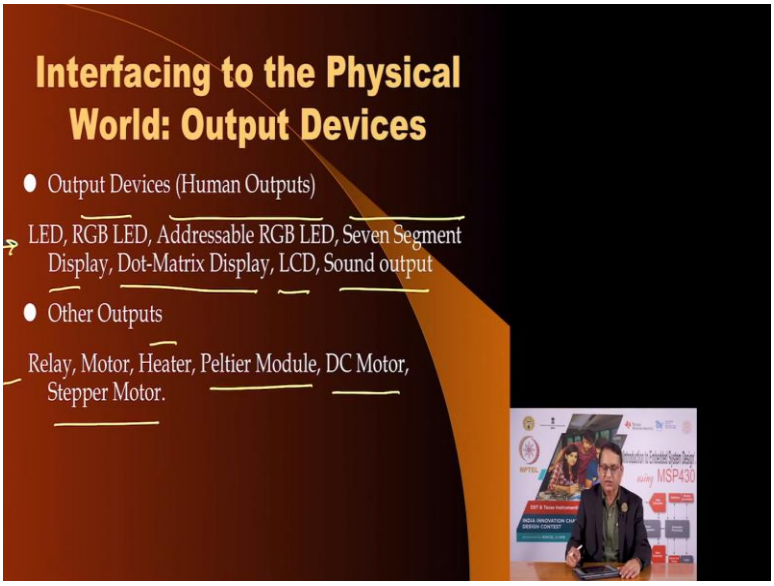
So, by measuring the relative phase of A and B you can know that the Rotary Encoder is being moved in a clockwise direction. How many times? What angle it has moved each pulse? So, each of this pulse each of this edge indicates that it has moved about 20 degrees, 25 degrees as the case in this case of 4 bit encoder, however, if now the Rotary Encoder is moved in the counterclockwise direction.

Now, you see for the same reference of B, the signal on A is a falling edge. So, you had to see whether during the time that B signal is high, does the A signal have a transition from 0 to 1 that means it goes in the clockwise direction. If it is going from 1 to 0 that means, it is going in the counterclockwise direction, the angle will be measured by how much, by how much angle the encoders mood will be measured by counting these pulses.

So, in the software in the microcontroller will have to keep a count and it can increment or decrement that count based on whether the encoder is moved in the clockwise or counter clockwise direction and the number of pulses will decide whether the count goes up or goes

down depending upon whether it is moved in the clockwise or counter clockwise. This is the anti clockwise direction. So, this is one of the popular inputs with which users can interact with the microcontroller system.

(Refer Slide Time: 28:18)



Now, let us see how we can connect some output devices to our microcontroller system and these would form from the, the output box and we see the ways multiple ways in which microcontroller can interact with the outside world would be through LEDs of various types. As we had listed in that block you have LEDs, normal LEDs, you have RGB LEDs. You can also addressable RGB LEDs, you have 7 segment displays.

You have dot matrix displays LCD and through sound output. Other outputs could be to control relay and motors you would like to control heaters or a Peltier module with which you can heat or cool you may want to control the speed and direction of SP DC motor as well as a Stepper motor. And so in this session we are going to in this part of the lecture we are going to consider how to control LED is to begin with.

And then we will see how we can control Relays, how we can control Peltier modules and DC motors and so on. We will resume this lecture, this discussion in the next lecture. Thank you very much.