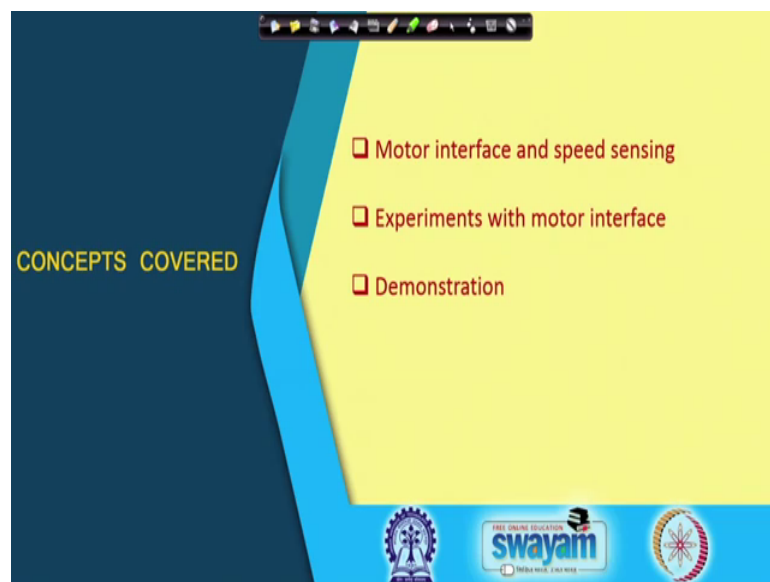


Embedded System Design with ARM
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 32
Experiments on Speed Control of DC Motor

In this lecture, we shall be showing you some more interfacing into experiments; namely we shall be showing you how to interface a DC motor with the STM 32 microcontroller board. We have used a small DC motor as an example to show you how this kind of interface can be done. So, we shall be showing you a couple of experiments on the controlling of the motor and how the speed can be varied, we shall be talking about that.

(Refer Slide Time: 00:51)

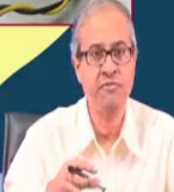
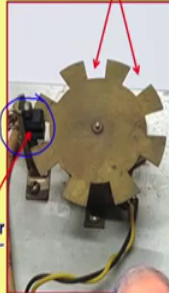


So, we shall be discussing that how motor can be interfaced and how speed can be sensed; we shall be talking about some experiments and demonstrations.

(Refer Slide Time: 01:03)

About the Motor Interface and Speed Sensing

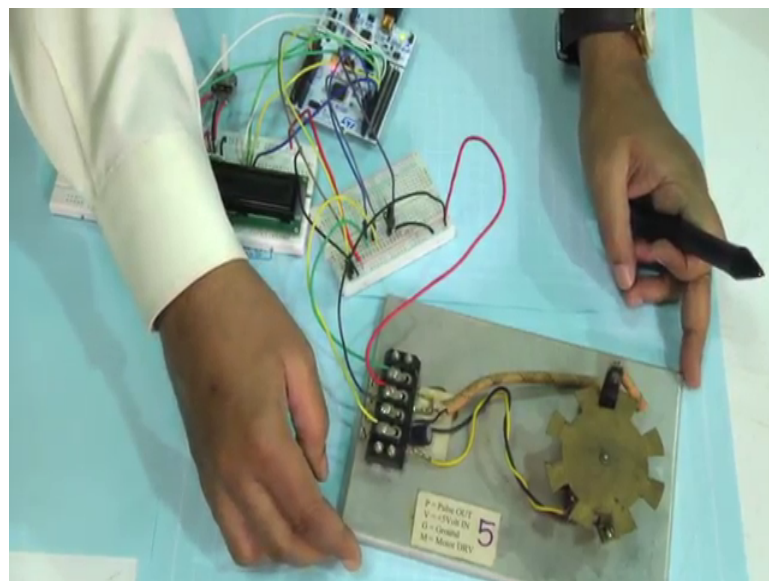
- A DC motor is used, where the rotating shaft is connected to a metal wheel cut with 8 slots.
- The wheel is attached to an optocoupler circuit, which generates optical interruptions whenever the wheel rotates.
 - 8 optical interrupts for every single rotation of the wheel.
 - An optocoupler circuit generates a pulse for every interruption.
- How is the motor driven?
 - Directly from the PWM port of the microcontroller.
 - By changing the PWM duty cycle, the speed of the motor can be varied.



THE ONLINE EDUCATION swayam

Now, the motor interface that I shall be showing you in this experiment.

(Refer Slide Time: 01:17)



Well, if you can see it once, this is the motor that I have interfaced. You can see there is a motor down below and there is a wheel which is connected to the shaft, which can rotate right this is the kind of setup I have used.

Coming back to this slide: so in the picture on the right, I have showed that same shaft what you can see that there are 8 slots, which are cut 1, 2, 3, 4, 5, 6, 7, 8. So, the rotating shaft is connected to the metal, wheel where there are 8 slots. And, on this side you can

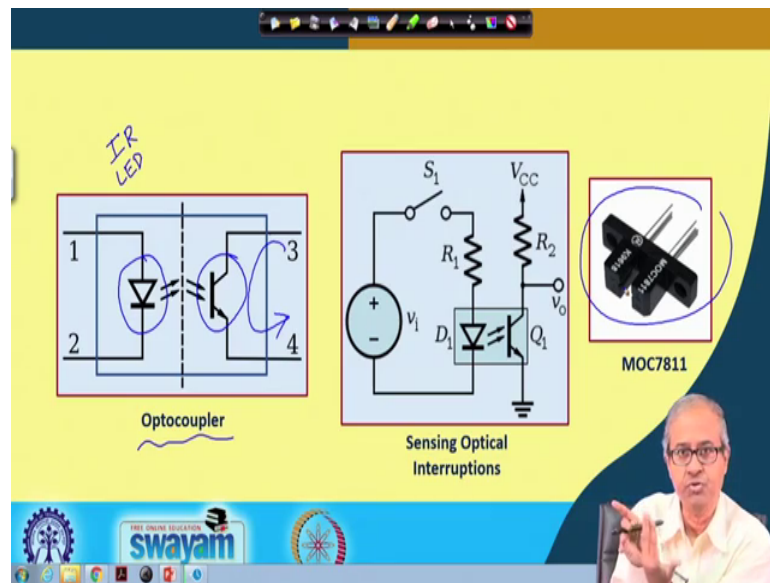
see a black kind of a thing there is an arrangement, which is an opto coupler circuit, which I shall be talking you about telling about. You see, earlier you had seen how an LDR circuit can be used to sense optical interruptions and count the number of objects, which are crossing a certain point may be number of persons entering and leaving a room.

Here we could have used LDR and some kind of LED light source also, but here we thought let us use some other kind of an optical; means sensing circuit, where you can also measure interruptions, and this is something which is called an optocoupler. So, we shall show you or tell you that how an optocoupler works, but the idea is that you see the optocoupler is strategically placed just in the place, where the wheel is rotating because, there are 8 slots in the wheel for one complete revolution, there will be 8 interruptions ok.

So, there will be 8 such optical interruptions for every rotation of the wheel. And this circuit that we will be using will be generating one pulse for every interruption for one revolution there will be 8 pulses. And the motor, because it is a small motor, we are using we can drive it directly from a microcomputer output port micro controller output port, we have used one PWM port. But, if it is a larger motor of course, you need to use a driver circuit, which can supply the required voltage and current to drive that larger motor, but this being a small motor we can drive it directly from the microcontroller port.

And the point is that because, we have used the PWM port again by changing the duty cycle the average value of the control that you are sending to the motor is changing. So, means we are turning on and off, on and off, on and off, so the average speed of the motor can be controlled. This is the idea, right.

(Refer Slide Time: 04:19)

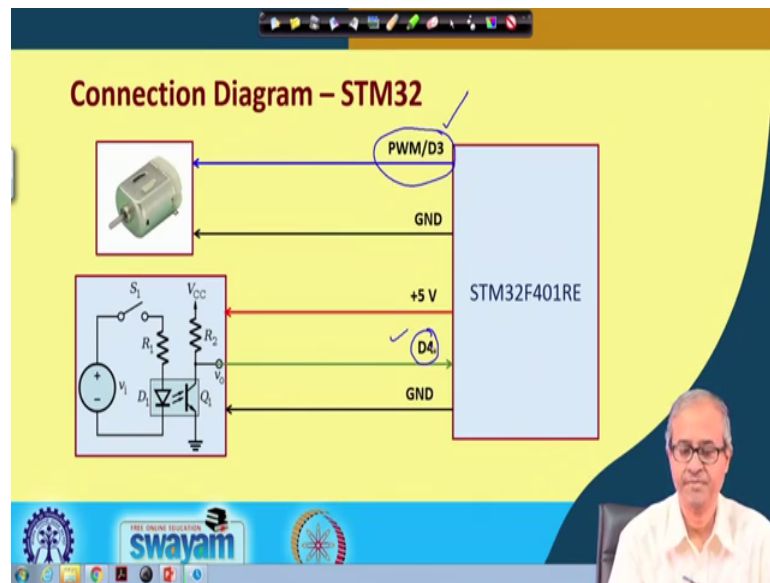


So, let us see how this optical interruption works using this optocoupler circuit. You see basically an optocoupler is the arrangement of two things: one is some kind of a light source, well in the circuit that we are using this light source is an infrared light emitting diode it is an IR LED. So, it emits a light, but in the infrared frequency band. And on the other side, there is a photo transistor, a photo transistor is a kind of a transistor, where there is no base connection like in a normal transistor, there are two terminals only. But the base is replaced by an optical mechanism, whenever there is a light the transistor turns on and when there is no light, the transistor is off.

So, whenever this LED throws a light on this transistor, the transistor conducts and there will be a current flowing path from 3 to 4, right. So, the circuit that you can use is something like this in the LED part there will be some kind of an LED driving circuit as you know, it will consist of a resistance in series with a power supply and this you can switch on and off as required. And on the other side, there will be another circuit using a resistance and this transistor, whenever the transistor is conducting this output voltage will be ground low voltage, and when it is off this output voltage will be close to VCC.

So, this optocoupler mechanism looks like this. You see there is a small hole in between and the window is actually placed in between those two plates here, these black plates right. This is how it is arranged.

(Refer Slide Time: 06:22)



Now, talking about the connection diagram the connection diagram is fairly simple. This motor desert said it will be driven directly from a PWM output port. So, here we have used the D port D 3 of course, there is another terminal to connect to ground. And for driving this motor or this speed sensing, this optocoupler circuit, I am using this 5 volts and ground because VCC and ground are required. And this output of this opto coupler, this is fed to digital input pin default, because this will be a digital output either 0 or 1, whether it is interruption or no interruption.

So, I am connecting it to D 3 for controlling the motor rotation and D 4 for sensing the optical interruption from the optocoupler. So, in our first experiment we shall be using only D 3, in the second experiment we shall be using both D 3 and D 4, right.

(Refer Slide Time: 07:31)

Experiment 1

- Interface the motor and a push switch, On successive presses of the switch, the following will alternate repeatedly:
 - a) Run the motor at high speed
 - b) Run the motor at low speed
 - c) Turn off the motor

The push switch is interfaced to port line D2.
PWM control is used to drive the motor.

The circuit diagram shows a +5V supply connected to a resistor, which is connected to a push switch. The other terminal of the switch is connected to port line D2. A hand-drawn checkmark is next to the D2 label.

swayam

So, let us first talk about the first experiment. In the first experiment, we interface the motor using the circuit that you have just now shown; in addition we are also interfacing a push button switch like this. So, if the switch is not pressed the output point will be high, if the switch is pressed it will be low; it will be connected to ground right. So, whenever switch is pressed the switch output, which is connected to port line D 2 will become 0, and see if the switch is not pressed, it will be 1.

So, the experiment goes like this. We would be continuously pressing the switches, initially the motor will be rotating at some speed. So, whenever we press one switch. So, there were 3 different levels of speed, one is the maximum speed, one is the medium speed and the other is off; motor will be turning off. So, when I go on pressing the switch maximum, medium, off: maximum, medium, off, this cycle will repeat right. This is how the experiment goes.

(Refer Slide Time: 08:49)

Embed C Code – STM32

```
#include "mbed.h"
PwmOut motor (D3) // D3 is set as a PWM controlled digital output
DigitalIn push_switch (D2) // Switch output is connected to pin D2
int main()
{
  int state = 1;
  motor.period (0.10);
  motor = 1.0; // Duty cycle = 1.0
  while(1) {
    if (push_switch == 0) { // Check for switch press
      state++;
      if (state == 4) state = 1;
    }
    while (push_switch == 0) // Wait for key release
  }
}
```

Handwritten annotations:
- Blue circles around (D3) and (D2) in the declarations.
- A blue arrow pointing from the comment // D3 is set as a PWM controlled digital output to the variable motor.
- A blue arrow pointing from the comment // Duty cycle = 1.0 to the assignment motor = 1.0.
- A blue arrow pointing from the comment // Wait for key release to the while loop.
- A blue handwritten note: motor.write(1.0);

Let us look at the program code, the program code is a little long it spans over 2 slides, let us see.

Here as I said, the motor control we are using this PWM control output line D 3 for controlling the motor, depending on the duty cycle the speed of the motor will vary. And the output of the switch push, switch that I said it is connected to D 2. This is a digital in type of input a digital input. In the main program here, we have defined a variable called state initialize to 1. State actually indicates that in which state the motor is because, it is in 3 states right maximum speed, medium speed and off.

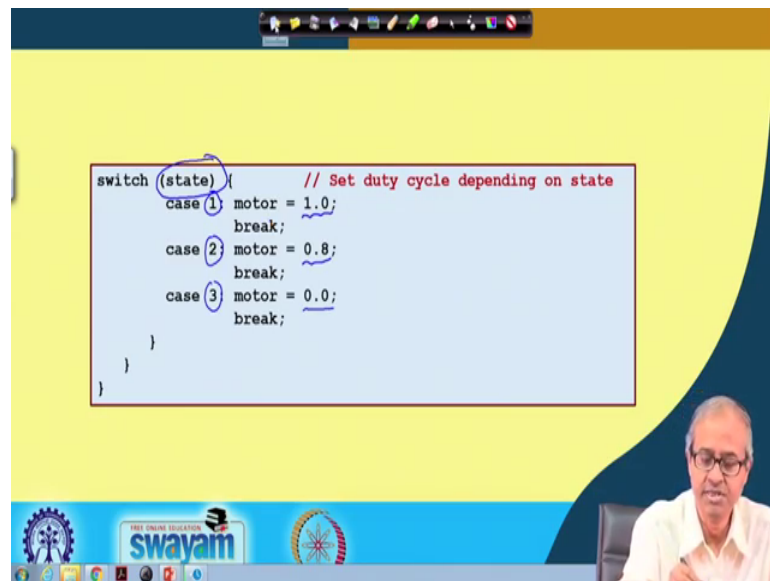
So, in which state it is that is indicated by this state variable; I am initializing it to one, it will go on 1, 2, 3, again 1 ok. Now, motor period, I have set it to 0.1 which means 100 millisecond. So, in 100 millisecond period, I repeat the process. And initially I set the duty cycle to maximum 1.0. Motor equal to 1.0, this is equivalent to motor dot write 1.0 in both these ways, we can set the duty cycle, you can either call this function write or you can simply write motor equal to 1.0 ok. So, initially it is rotating in the maximum speed.

Now, in this while loop first, we are checking for a switch press; if a switch is pressed so how do you check? If the push switch this digital in whose name is 0, 0 means switch has been pressed. If it is 0 then I increment state by 1 initially, it was 1, it will become 2 then I check if it has become 4, because it can be only 1, 2, 3, right. So, after 3 if you do plus

plus, it will become 4. See if it tries to become 4, you again bring it back to 1. So in this loop, it will go along 1, 2, 3 again 1, 2, 3 like this.

And after this switch press is detected and you have incremented state, you wait till the switch is released. So, as long as switch remains 0 I wait in a dummy loop; this is a dummy loop. So, if I if you keep the switch pressed for a long time, it will wait in this loop as you release it then only the motor speed next one will take effect.

(Refer Slide Time: 11:48)

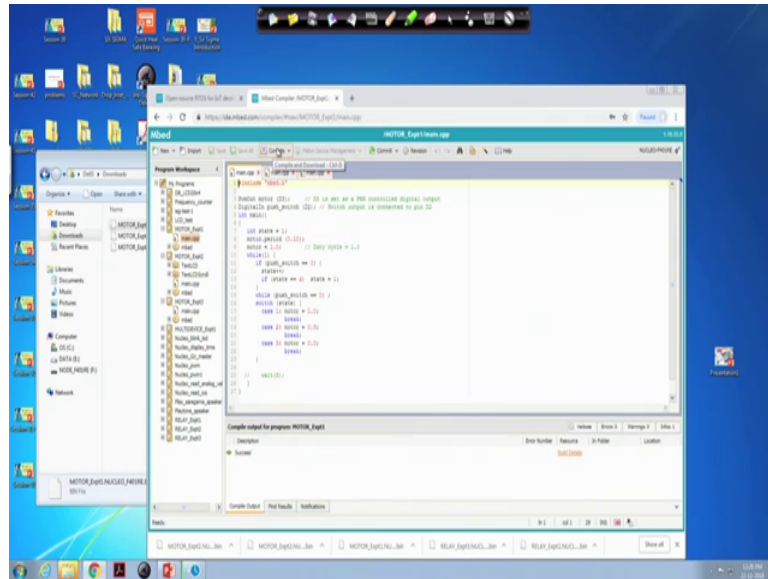


```
switch (state) { // Set duty cycle depending on state
  case (1) motor = 1.0;
            break;
  case (2) motor = 0.8;
            break;
  case (3) motor = 0.0;
            break;
}
```

So, the next slide we actually show you how the speed control is done. Then there is a switch case statement. There is a switch statement, which actually checks the value of state variable state, it can be either 1, it can be 2, it can be 3. So, if it is 1 then you are using the maximum duty cycle, if it is 2 the next one this is medium speed and if it is 3 then it is 0.0. So, this thing you go on repeating in a cycle, right.

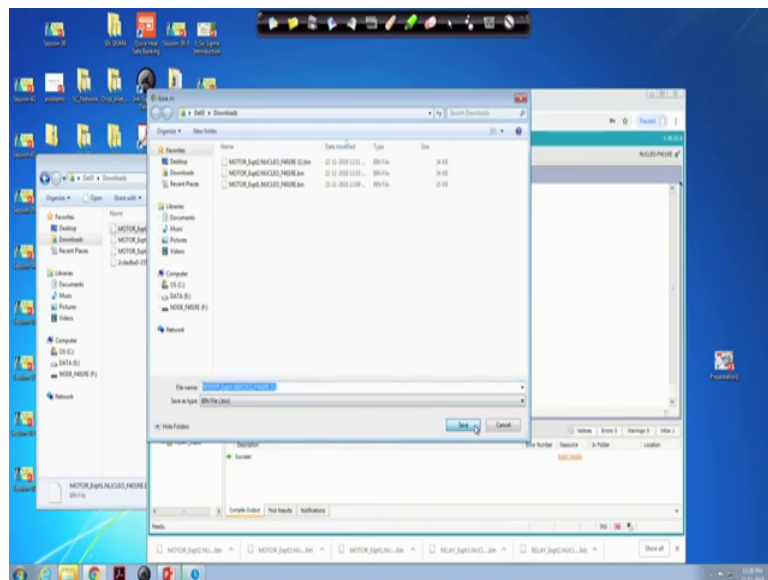
So, let us now see the demo ok.

(Refer Slide Time: 12:46)



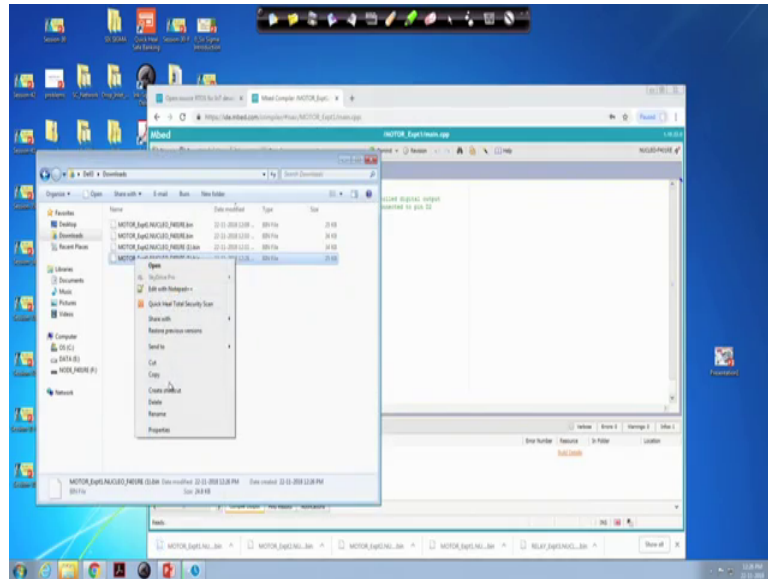
This is the first experiment, just the experiment that we have shown that same one, let us compile it.

(Refer Slide Time: 13:07)



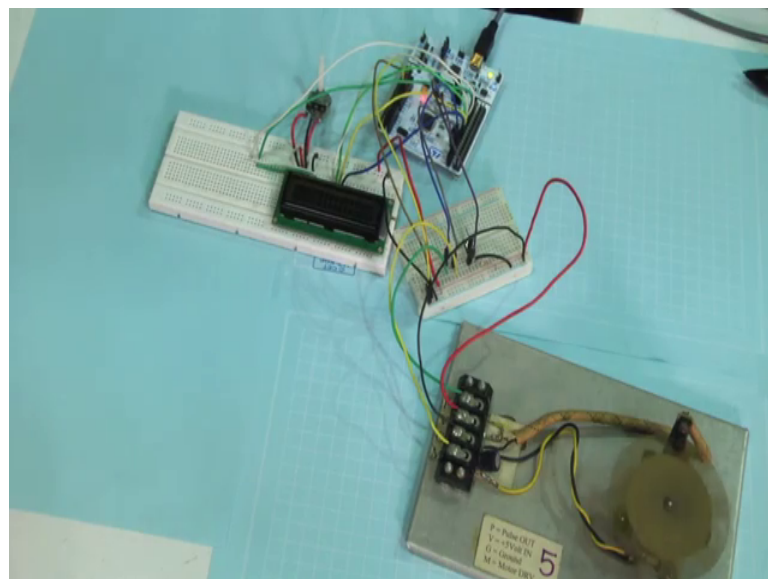
Save it.

(Refer Slide Time: 13:10)



Copy and paste.

(Refer Slide Time: 13:18)



Now see this motor starts rotating. So, you can see here, that the motor is rotating and it is the maximum speed and here I have interfaced a small push switch here. Here, you can see here the push switch and one terminal the push switch is connected to this resistance to VCC and the other terminal of the switch is connected to ground. And the middle point of the switch this one, this is connected to the input line D 2 ok.

And regarding the interface of the motor, if you see there are some terminals. This is the 5 volts, which are connected to the 5 volt point, this is black wire is ground, I have connected to ground, and this yellow wire is the control is the PWM control is the PWM control, which is connected to D 3 PWM D 3 line. And this other line green one, this we shall be using in the next experiment, this will be using the optocoupler output ok. But in this experiment we are not using this optocoupler, right.

So, you see motor is rotating in the full speed if I press this switch once I press once. Now the speed has decreased, now it is medium if I press another time motor will turn off ok. I press once mode it will again start rotating in the maximum speed, then I press once more medium speed ok. This is maximum speed, medium speed, off all right. This cycle will repeat.

Let us continue.

(Refer Slide Time: 15:29)

Experiment 2

- Interface the motor and read the pulses generated by the rotary sensor. Count the pulses and display the RPM on a LCD display unit.

Diagram showing a motor (M) connected to D3 and D4, and an LCD display unit connected to D3.

- The pulses generated by the optocoupler are fed to interrupt input D4.
- PWM control is used to drive the motor.
- LCD display unit is also interfaced.

Now the second experiment, we are doing certain things. Now you think of the motor, you have the motor out here; this is the symbol of a motor. Now normally what we have done, we are using one control line from the microcontroller to control the speed. So, that we have connected to the PWM output line D 3.

Now in this experiment, we are also using the output of the optocoupler because, we want to count how many or what is the speed of rotation, how many pulses are coming

per second something like that or per minute. So, the optical output we are connecting to digital input line D 4, right. Now after counting the point is how do I know, what is the speed, how do I show?

So for that reason, we have also interfaced an LCD with the microcontroller circuit. So, I am not showing the details of LCD interface, because you have already studied this and saw this in some earlier lecture. So, you know how LCD is interfaced, we have an interface and said in the 4 bit mode as was demonstrated earlier. So this is how it works.

Now this is another point I shall be explaining when I show you the program ok.

(Refer Slide Time: 17:09)

```
Embed C Code – STM32

#include "mbed.h"
#include "TextLCD.h"
#include "TextLCDScroll.h"

PwmOut motor (D3); // D3 is set as a PWM controlled digital output
InterruptIn wheel (D4); // Optocoupler output is connected to pin D4

TextLCDScroll lcd(D13, D12, D11, D10, D9, D8, TextLCD::LCD16x2);

int pulses = 0;

void count()
{
    pulses = pulses + 1;
}
```

Interrupt → D4 STM32

Now, one point you see. Now I here, as I have already this mbed dot h is included and for LCD interface, I need to interface textile city. Of course, we do not even scroll, so the third one is not required really, but I have used both. And motor drive, I have connected to D 3 PWM out and D 4 is my input that is coming from the output of the optocoupler.

Now, let us try to understand. When I write a program, how do I count? Suppose, I want to count, how many such pulses are coming per second, how do I count? There must be a way to count, how many pulses are coming per second. Now the way we have, implemented it in our program is that we have used the concept of interrupt, what is an interrupt? Interrupt is a mechanism like this; suppose, we have a processor this is our

STM 32, this is our processor board and from outside some interrupt is coming. Here, we have connected it to the digital input line D 4.

Now, when you discussed interrupts earlier, you recall you told you that any of the digital IO lines can be used as an interrupt pin. So, here we are using this D 4 as an interrupt input. How do I clear that? I declare it as interrupt in, instead of telling it digital in, I mention it as interrupt in, and the name of this object, I have given as wheel and for PWM out I am calling it motor, right.

Now, the point is that for interrupt what really happens? Means, if you have studied it earlier in some course or somewhere you know that when an interrupt signal comes to a processor, whatever the processor was executing that is temporarily suspended and the control jumps to another programmer routine that is called interrupt service routine. After execution of the interrupt service routine control comes back to the program which were suspended. This is how it works.

Now in this case, every time there is an optical interruption. So, for one revolution, there will be 8 such interruptions. So, there will be some pulses coming like this ok. So, every time there is an optical interruption, there will be a pulse and each of these pulse will be generating an interrupt signal. So, interrupt service routine will be called so many times ok.

(Refer Slide Time: 20:05)

```
#include "mbed.h"
#include "TextLCD.h"
#include "TextLCDScroll.h"

PwmOut motor (D3); // D3 is set as a PWM controlled digital output
InterruptIn wheel (D4); // Optocoupler output is connected to pin D4

TextLCDScroll lcd(D13, D12, D11, D10, D9, D8, TextLCD::LCD16x2);

int pulses = 0;

void count()
{
    pulses = pulses + 1;
}
```

The slide also features handwritten annotations: 'RS' and 'EN' with arrows pointing to pins D13 and D12 respectively; 'Data' with an arrow pointing to pins D11, D10, D9, and D8; and a circuit diagram showing a pull-up resistor connected to +5V and pin D4, with the other end connected to VEE.

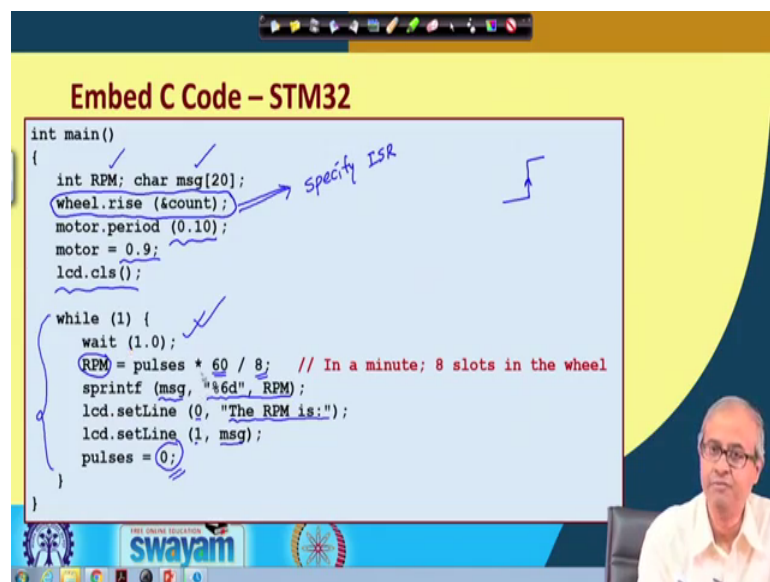
So, what we have done here is that. See, this is actually the interrupt service routine, this count this function, I will show you how we declare it as an interrupt service routine. But, suppose for the time being this is the interrupt service routine, what I have done: we have declared a global variable integer variable called pulses initialize it to 0. And in the interrupt service routine, we are just increasing pulses by 1 pulse is equal to pulses + 1,; so that whenever interrupt comes the variable gets incremented by 1.

And the other thing to note here is that as it said we have also interfaced an LCD and these are the standard interfaces, you recall the first of these is your that register select, second one is your enable, and last 4 are the most significant data lines for 4 bit interface.

So, we have connected the pins from the LCD to D 8, D 9, D 10, D 11 and this enable to D 12 and RS to D 13. And of course, VCC ground and that you recall there is a potentiometer which we used like this, this we connect it to another pin called VEE for LCD contrast arrangement, but this was connected to 5 volts. So, this potentiometer circuit is also there. So, this LCD circuit I have already done, but I am not showing it here because, you already have seen earlier how LCD is interfaced ok.

Now I told you this is the interrupt service routine.

(Refer Slide Time: 21:53)



```
int main()
{
    int RPM; char msg[20];
    wheel.rise (&count);
    motor.period (0.10);
    motor = 0.9;
    lcd.cls ();

    while (1) {
        wait (1.0);
        RPM = pulses * 60 / 8; // In a minute; 8 slots in the wheel
        sprintf (msg, "%6d", RPM);
        lcd.setLine (0, "The RPM is:");
        lcd.setLine (1, msg);
        pulses = 0;
    }
}
```

Now, let us see how this interrupt service routine is mentioned. This is our main function. First thing is that you note this line this is the place where we are specifying the interrupt

service routine. We are saying that wheel is that object, which is connected to that interrupt line that we called wheel, rise is a function means it checks every time the signal rises from 0 to 1 that is called rise, whenever there is a rise you call this function ampersand count means this is a pointer to a function. That count was the name of that function, you call count every time there is a rising edge on this wing.

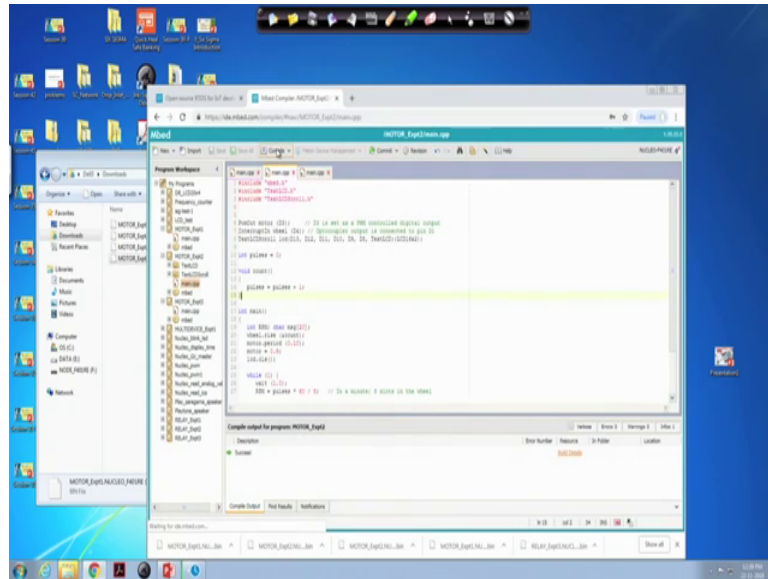
That is how interrupt service routine gets called and we have declared a variable called RPM, where you are counting revolutions per minute. There is a character array called mems g message. For the motor driving, we have assumed a period of 100 milliseconds as in the early experiment same thing, we chose and for motor initially the duty cycle is set to 0.9 and cls function initially clears the LCD screen. And in the main program main, while loop what we do.

See is initially the pulse s was initialized to 0, now we wait for 1 second wait 1.0; that means, in this 1 second the motor is rotating. So, there how many pulses are coming in this one second will get counted pulses will get incremented by so many times. At the end of it we calculate revolutions per minute because, we have counted for 1 second, for 1 minute, it will be multiplied by 60, and because there are 8 holes in that wheel 8 slots. So, for every revolution there will be 8 interruptions. So, we have to divide it by 8. This will give you your revolutions per minute or RPM value.

And what we do we sprint f means you can print this value into a string, you are printing the value of this RPM as an integer into this corrector string. Then, we are displaying on LCD it is a 2 line insulin display recall. on line number 0 we are displaying a string the RPM is, and on line number 1 we are displaying this message whatever is the RPM.

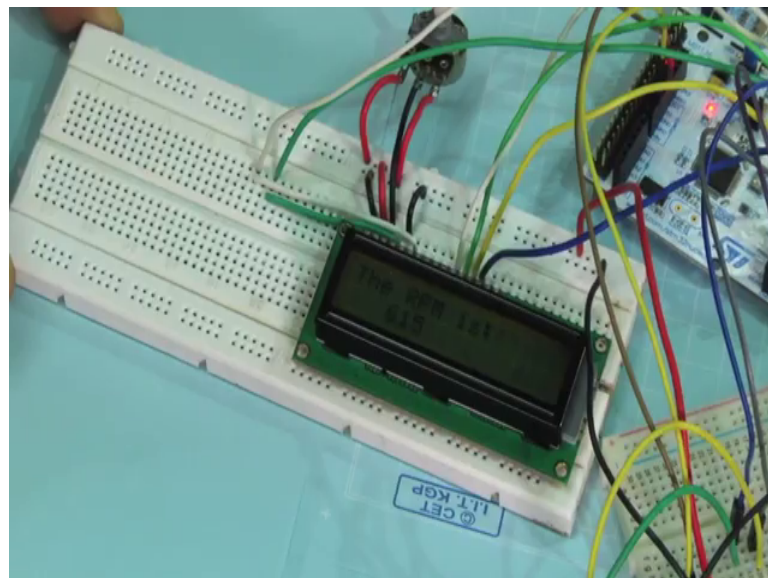
And after we do this we reinitialize the value of pulses to 0 so that in the next cycle we again carryout with the counting and you repeat this process. Again wait for 1 second, again the pulses will get counted for 1 seconds, again you calculate RPM and print it ok. This is the program, and let us see the demo now.

(Refer Slide Time: 25:14)



So, here this is our experiment 2. So, this is the same program that I have shown the same code, this I am trying to compile. So I compile it, save it, copy it and paste it to a full. Now, let us see what is happening.

(Refer Slide Time: 25:50)



See, in the program we have sent the duty cycle to 0.9. So, see the motor is rotating at a very high speed, resilient high speed. Now the switch circuit is not being used in the experiments. So, this switch is not read, but you look at this LCD display here.

This is the potentiometer for controlling the contrast, so you can adjust it for a suitable contrast. If it is showing the RPM is 1095 1065, you see this motor is a very cheap and small motor, so this speed is not very stable. So, it is varying a little bit, so 1027, 1100 1095. So, the RPM is going like this, right. So, this you can view this RPM value here, this RPM value you can see right ok.

Now let us see if I change the duty cycle does the RPM value changes, let me see this. You see here, the motor duty cycle was set to 0.9. Let me change it to 0.9 to 0.7 let us say, let me change it to 0.7. Let us save it, compile it again and we save this, copy, paste. So now, see the same thing is happening, but now the motor is rotating at a much slower speed. If you look into the motor, it is a little slower it is rotating at a slower speed. And now, if you display, you see the RPM value displayed it is showing 600 something 592, 600, 10, 15 like that. So, the RPM values are dropped significantly you see.

So now, you can see that if you change the duty cycle of the signal that is used to drive this motor ok. So, the RPM value automatically changes and how you are doing? This green line is the output of this optocoupler, this is connected to this interrupt input pin default and as the program has been shown, we are counting that multiplying by 60 and then dividing it by 8 to get the value of the revolutions per minute, which is then displayed on this LCD screen, which you can see out here. This is the overall experiment ok. So now you see, just one thing let me tell you in this context. Suppose I ask you to modify this program to control the speed of a motor, so you can do it very easily. Suppose, I say you set the RPM value to 800. So, you do not know how much duty cycle will make it 800.

So, in this way you calculate, read the value, and if you find that the RPM value is lower, you increase the duty cycle a little bit and again read; if it is lower you again read. See earlier, we talked about this integral control on off control, PID control all these things, so all these things can come into the picture here. But, because the program will become complex, you are not showing those in this demonstration, but you can also have it; you can set a preset RPM value, and you can adjust the RPM adjust the motor drive to make it work accordingly.

With this we come to the end of this lecture, thank you.