



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Dr. Kamalika Datta

Department : Computer Science and Engineering

Topic

**Lecture 24: Serial Port Terminal Application
(Coolterm)**

CONCEPTS COVERED

☐ About CoolTerm

☐ Configuration

☐ Demonstration



Introduction

- Sensor data depends on the connection as well as on noise so we need to check the value of sensor.
- The value must be printed on the screen.
- To print the value on the PC screen we need a communication channel.
- A serial communication link between the PC and microcontroller is used to transfer data from microcontroller to PC.
- Software tools such as *CoolTerm*, *TeraTerm*, etc. are used to establish the serial communication and to print the data on the PC screen.

HyperTerminal

- Software Tools that enables serial communication between the desktop/laptop and the microcontroller are termed as HyperTerminal Tool.
 - Examples: CoolTerm, TeraTerm, etc.
- For our experiments, it is required to ensure that there is some HyperTerminal installed on the computer.
 - Will enable serial data communication with the microcontroller.
 - HyperTerminal connections in the present context shall be made using USB port.

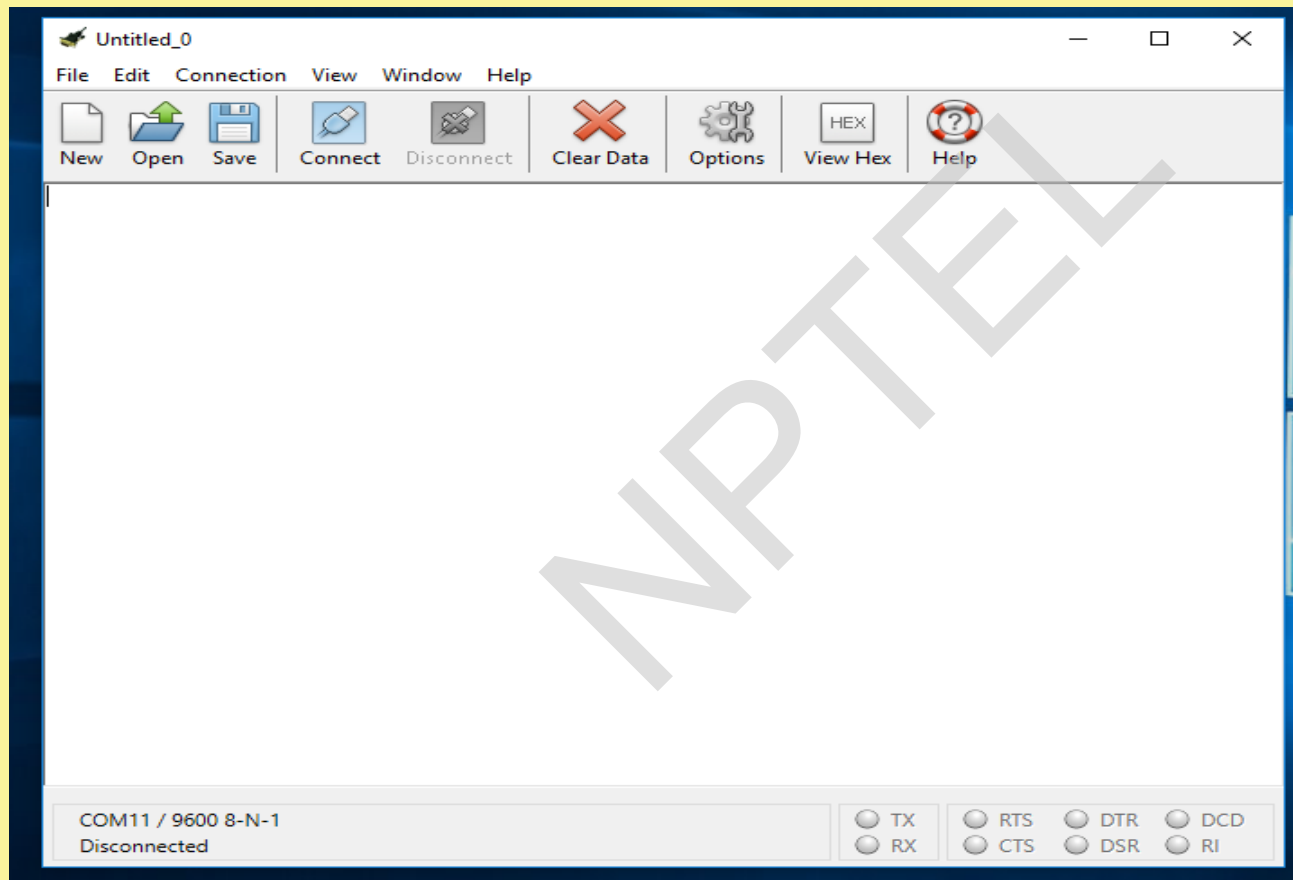
CoolTerm

- In our experiments we shall be using *CoolTerm*.
 - Available on a wide variety of platforms.
- It can be downloaded from several sites:
http://download.cnet.com/CoolTerm/3000-2383_4-10915882.html
<http://coolterm.en.lo4d.com/>
- After extracting the file we shall get the *CoolTerm.exe* file.
- This will be used for serial connection between the PC/laptop and the STM32 kit.

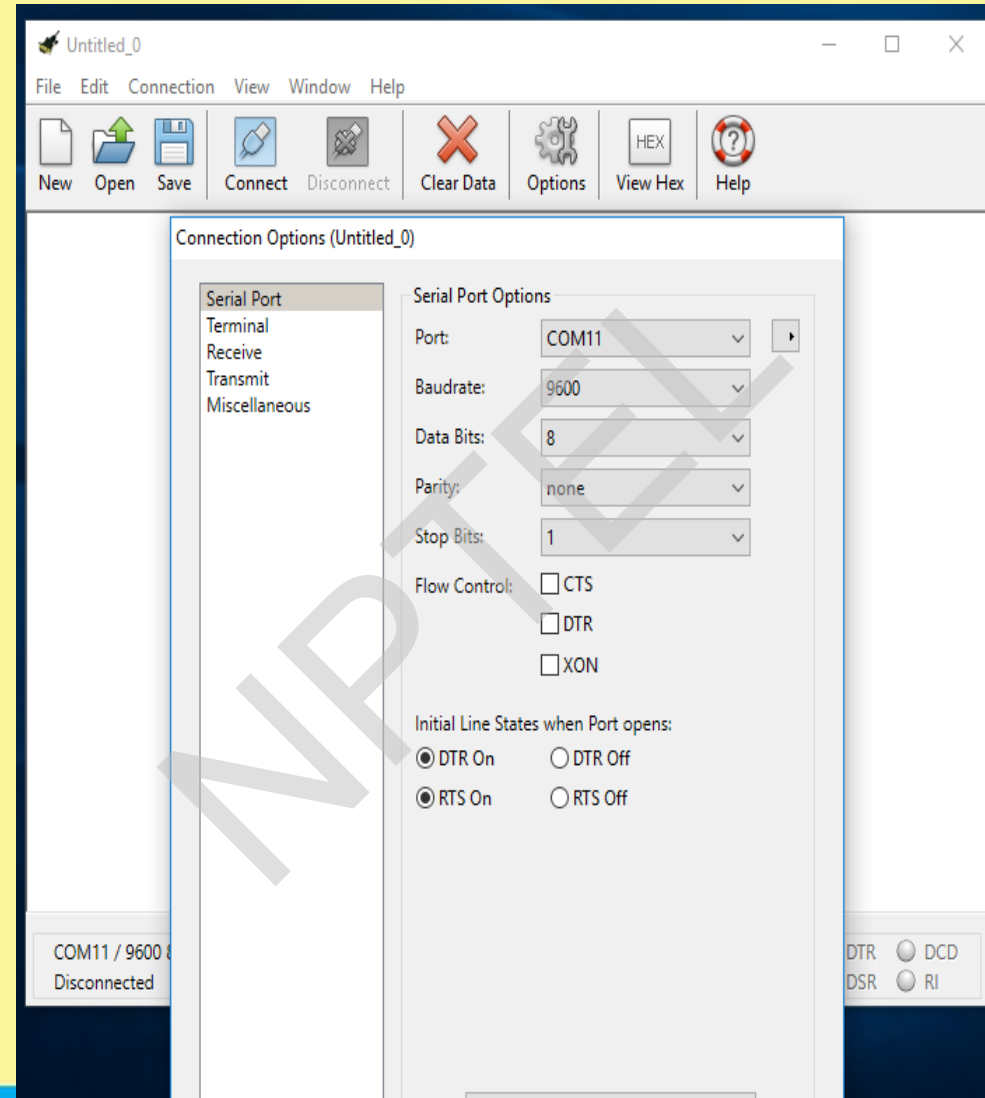
Settings on CoolTerm

- For proper data communication, proper settings have to be made on the HyperTerminal software.
 - We shall explain the steps for CoolTerm.
- The default serial configuration in CoolTerm is 9600 bauds, 8-bit data.
- If you have not set any bauds rate in the program, then no need to change on this. Else it can be set to match with the program.

Connect



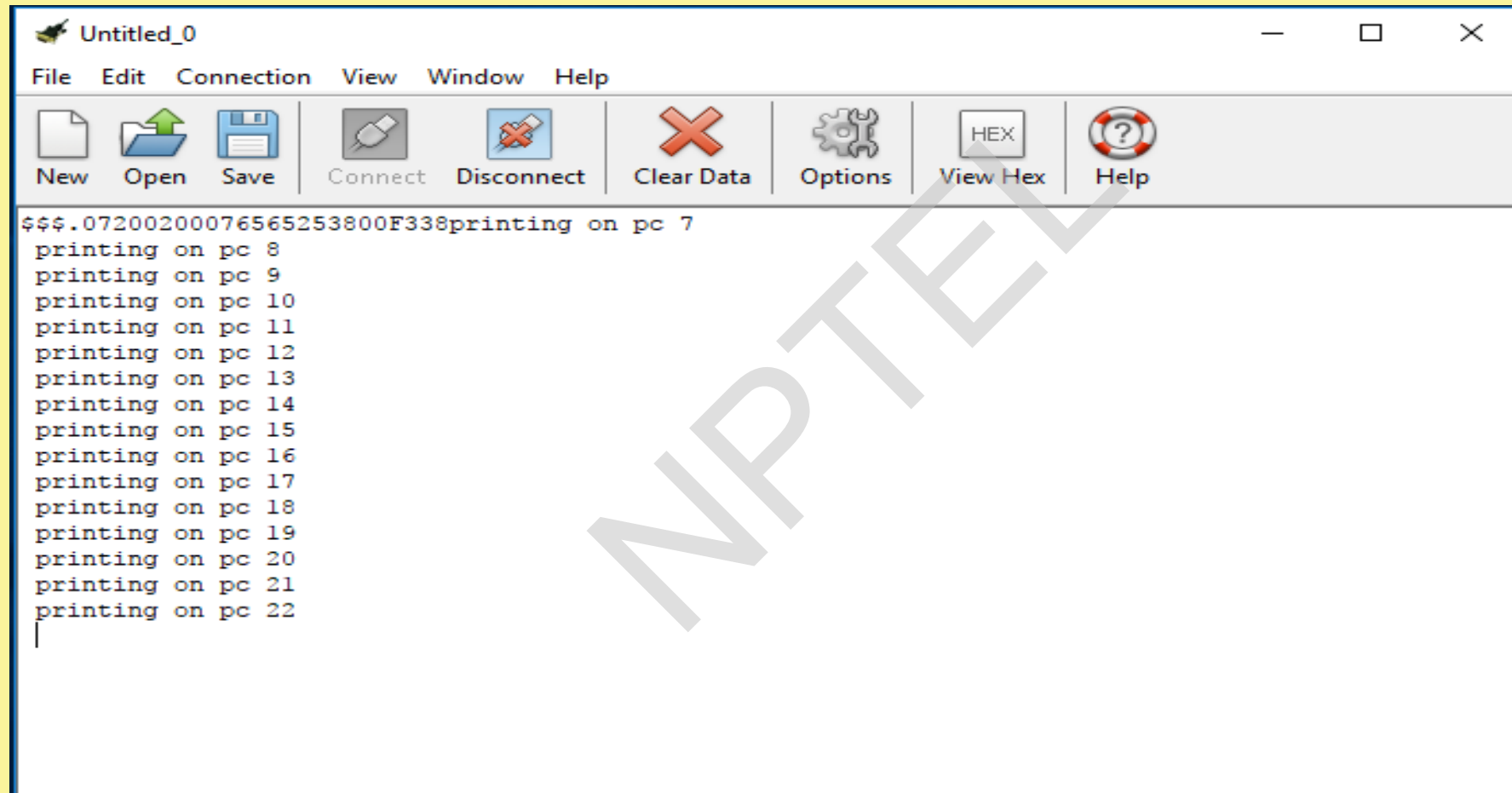
Configuration



Example

```
main.cpp
1 #include "mbed.h"
2 Serial pc(USBTX, USBRX);
3 int i=0;
4 int main(){
5     while(1) {
6         pc.printf("printing on pc %d \n ", i++);
7         wait(1.0); //wait for 1 second
8     }
9 }
10
```

Output



The screenshot shows a serial terminal window with the following content:

```
$$$$.07200200076565253800F338printing on pc 7
printing on pc 8
printing on pc 9
printing on pc 10
printing on pc 11
printing on pc 12
printing on pc 13
printing on pc 14
printing on pc 15
printing on pc 16
printing on pc 17
printing on pc 18
printing on pc 19
printing on pc 20
printing on pc 21
printing on pc 22
|
```

The window has a menu bar (File, Edit, Connection, View, Window, Help) and a toolbar with icons for New, Open, Save, Connect, Disconnect, Clear Data, Options, View Hex, and Help.



NPTEL ONLINE CERTIFICATION COURSES

Thank
you!



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Dr. Kamalika Datta

Department : Computer Science and Engineering

Topic

Lecture 25: Experiment with Temperature Sensor

CONCEPTS COVERED

- ☐ Interfacing LM35 to STM32 board
- ☐ Calibration and program code
- ☐ Demonstration

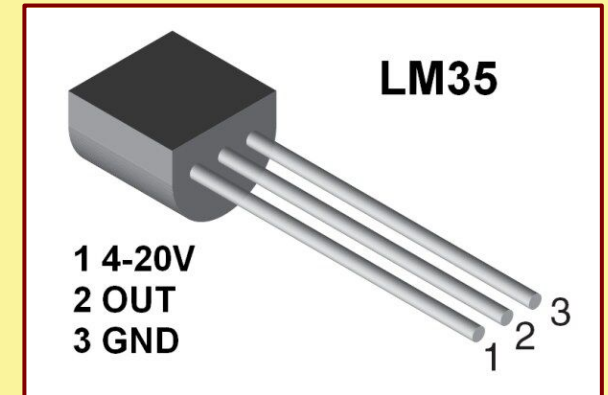


Temperature Sensing

- Sensing temperature has various applications:
 - Measure temperature of body, sophisticated Industrial appliances, chemical plants, etc.
 - Weather forecast.
- Thermometer is the most widely used instrument for temperature sensing.
 - A mercury thermometer does not produce signals on temperature change, instead changes its property (viz. expansion or contraction).
 - A digital thermometer displays temperature on a tiny LCD display.

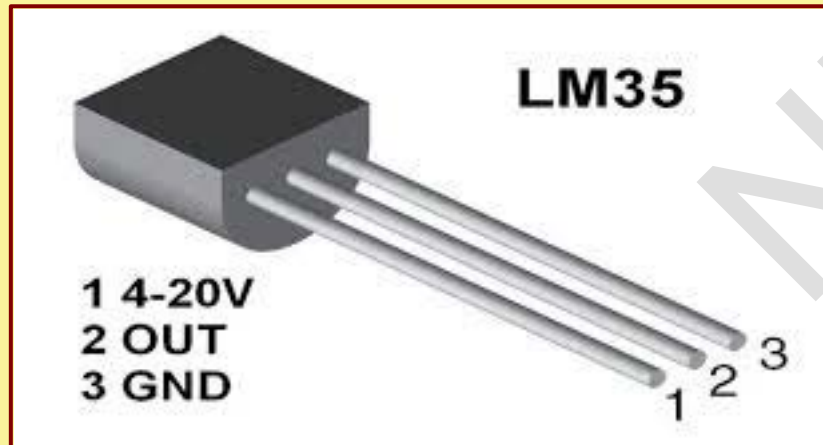
LM35 Temperature Sensor

- Precision integrated-circuit temperature device with an output voltage linearly proportional to the temperature in degrees Celsius (or Fahrenheit).
- Linear characteristic: $+10 \text{ mV}/^{\circ}\text{C}$ change in output.
- Rated for full -55°C to 150°C range.
- How to use the LM35?
 - Connect power supply (say, +5V and GND) to pins 1 and 3.
 - Measure the analog voltage output on pin2.
 - We can connect pin 2 to one of the analog input pins of the microcontroller.



Reading Analog Value from LM35

- Connect the two extreme terminals to power (say, 5V) and GND.
 - The middle terminal will produce a voltage proportional to the temperature.



```
#include "mbed.h"
AnalogIn sensor(A1);
...
float p;
p = sensor.read();
...
pc.printf ("\n Value read: %f", p);
...
```

Calibration

- LM35 output voltage increases linearly with increase in temperature.
 - 10 mV increase for every $^{\circ}\text{C}$ rise in temperature.
 - For $k^{\circ}\text{C}$, the output voltage will be $10k\text{ mV}$.
 - If maximum temperature is 50°C , maximum output voltage will be 0.5 V ($10 \times 50 = 500$).
- How to compute the temperature in $^{\circ}\text{C}$?
 - The *read()* function in *AnalogIn* class returns a fraction between 0 and 1.
 - Apply a voltage 0.5V directly to the analog input pin, and measure the value printed by the program (suppose, the value printed is P_{max}).
 - For an unknown temperature, if the value printed is P , the temperature value can be calculated as

$$T = 50 / P_{max} * P$$

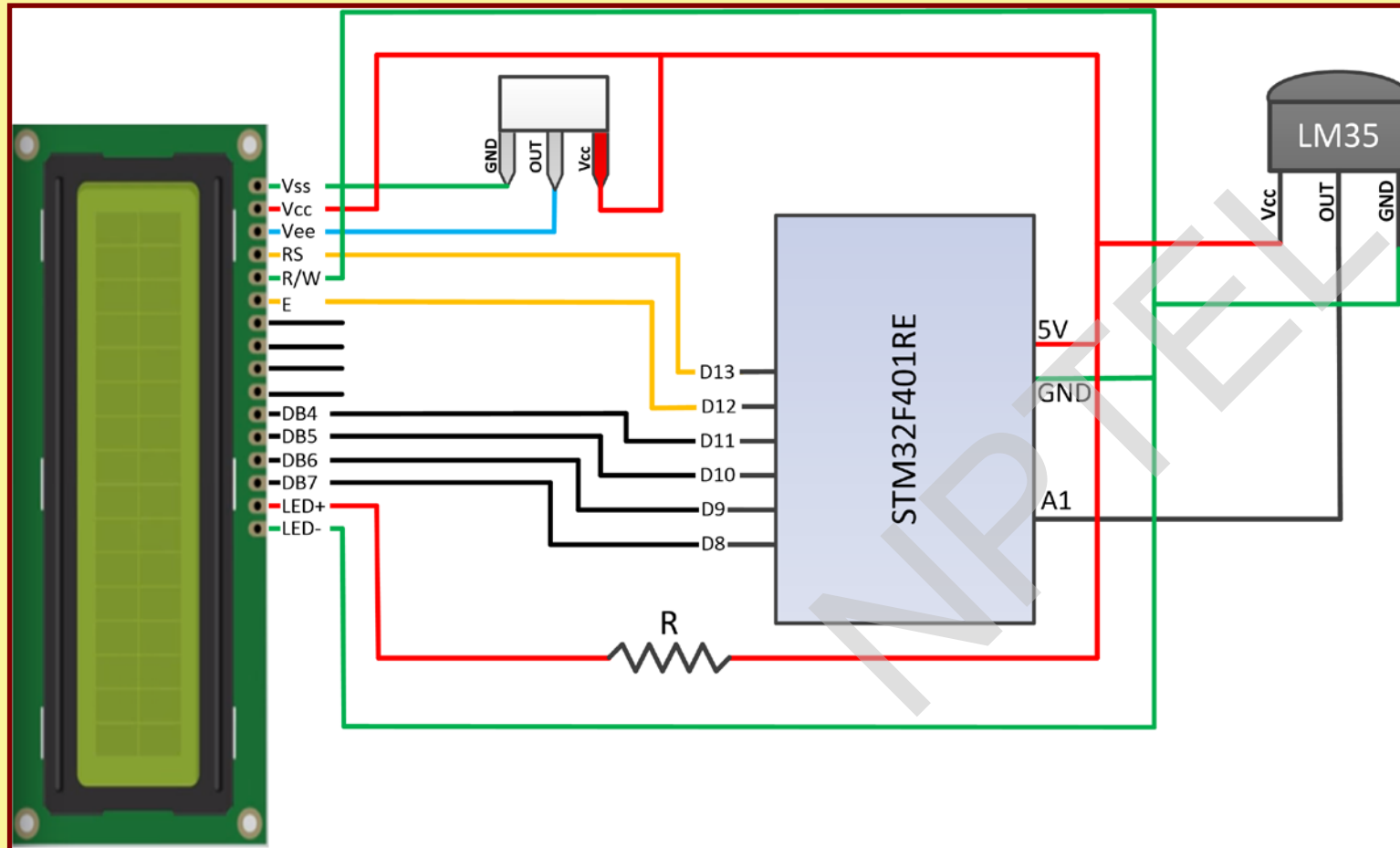
- Alternate way of computing:

- Suppose the room temperature is 20°C
- Using Coolterm, we run the program and suppose we find the value printed as 0.06
- For any unknown temperature, if the value printed is P , the temperature in $^{\circ}\text{C}$ will be $20P / 0.06$

An Experiment using LM35

- Design a *Temperature Sensing Unit* using STM32 Nucleo-F401RE kit and a LM35 temperature sensor to sense the environmental temperature, and display it in Centigrade scale on a LCD display unit.
 - The analog output voltage from LM35 is connected to pin A1 of the Arduino connector of the STM32 board.

Connection Diagram



Mbed Program Code

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

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

Serial pc (USBTX, USBRX);
AnalogIn sensor(A1);
DigitalOut led(D2);
```

```
int main(){
    int val=297; // after calibration
    char buf[20];
    float p;
    lcd.setLine(0, "Temp in Degree C");
    while(1) {
        p = sensor.read();
        sprintf(buf, "%4.2f", p*val);
        lcd.setLine(1, buf);
        wait(1);
    }
}
```



NPTEL ONLINE CERTIFICATION COURSES

Thank
you!



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Dr. Kamalika Datta

Department : Computer Science and Engineering

Topic

Lecture 26: Experiment with LDR Light Sensor (Part 1)

CONCEPTS COVERED

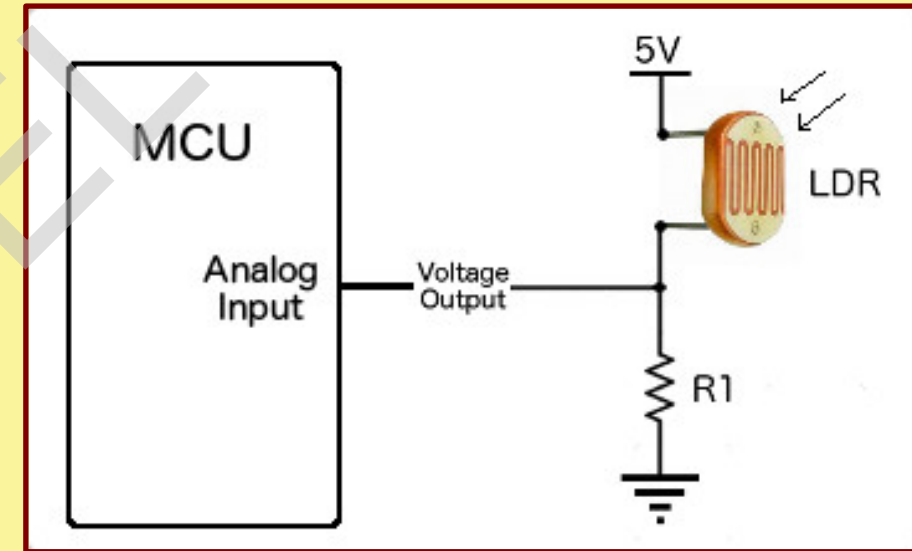
- ☐ LDR interface circuit
- ☐ Experiment for sensing ambient light level and changing intensity of a LED (using PWM)
- ☐ Demonstration



Light Dependent Resistor (LDR)

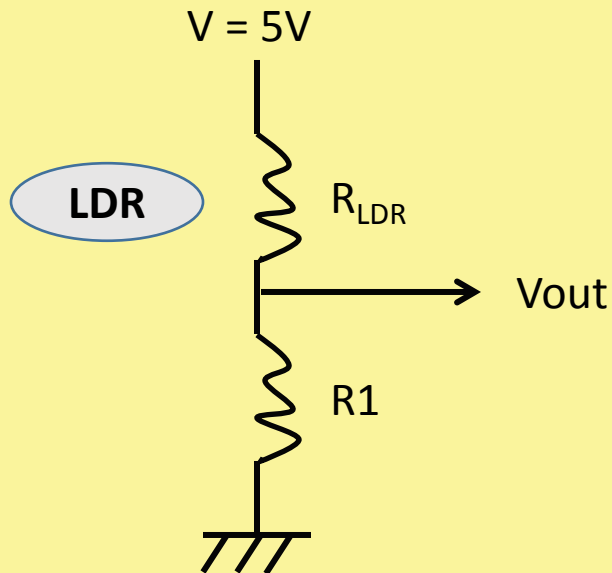


- A LDR is a variable resistor (a passive component) whose resistance value changes depending upon the amount of light falling on it.
 - More the amount of light, less will be the resistance and vice versa.
 - The variation of resistance with light intensity is non-linear.
- A simple resistance divider can generate an analog voltage that depends on R_{LDR} .



LDR Interfacing

- First test the LDR by measuring its resistance.
 - Consider for example, $R_{LDR} = 150\text{ K}\Omega$ (no light), and $R_{LDR} = 10\text{ K}\Omega$ (light)



$$\begin{aligned} I &= V / (R_{LDR} + R1) \\ V_{out} &= I * R1 \\ &= V * R1 / (R_{LDR} + R1) \\ &= 5 * R1 / (R_{LDR} + R2) \end{aligned}$$

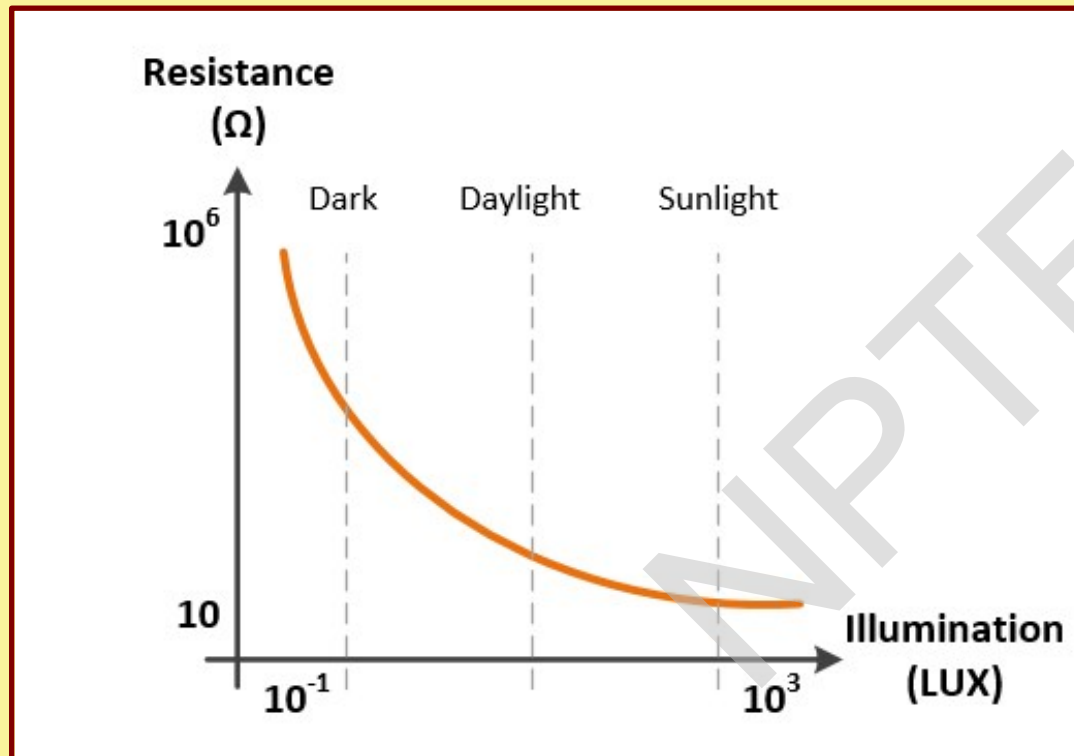
How to choose R1?

- Within the range of light intensity for some application, the variation in output voltage should be appreciable.
- Consider for example, $R_{LDR} = 150\text{ K}\Omega$ (no light), and $R_{LDR} = 10\text{ K}\Omega$ (light)

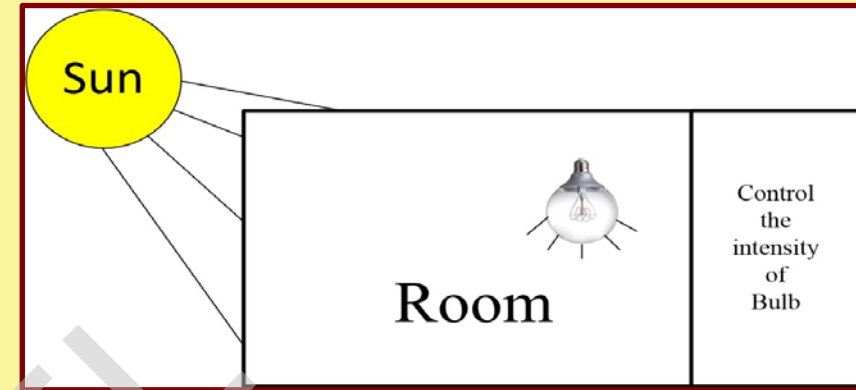
$$V_{out} = 5 * R1 / (R_{LDR} + R1)$$

R1	V _{out} (light)	V _{out} (no light)
1 K Ω	0.45 V	0.03 V
5 K Ω	1.67 V	0.16 V
10 K Ω	2.50 V	0.31 V
20 K Ω	3.33 V	0.56 V
50 K Ω	4.17 V	1.25 V

Nonlinear Variation of Resistance with Light

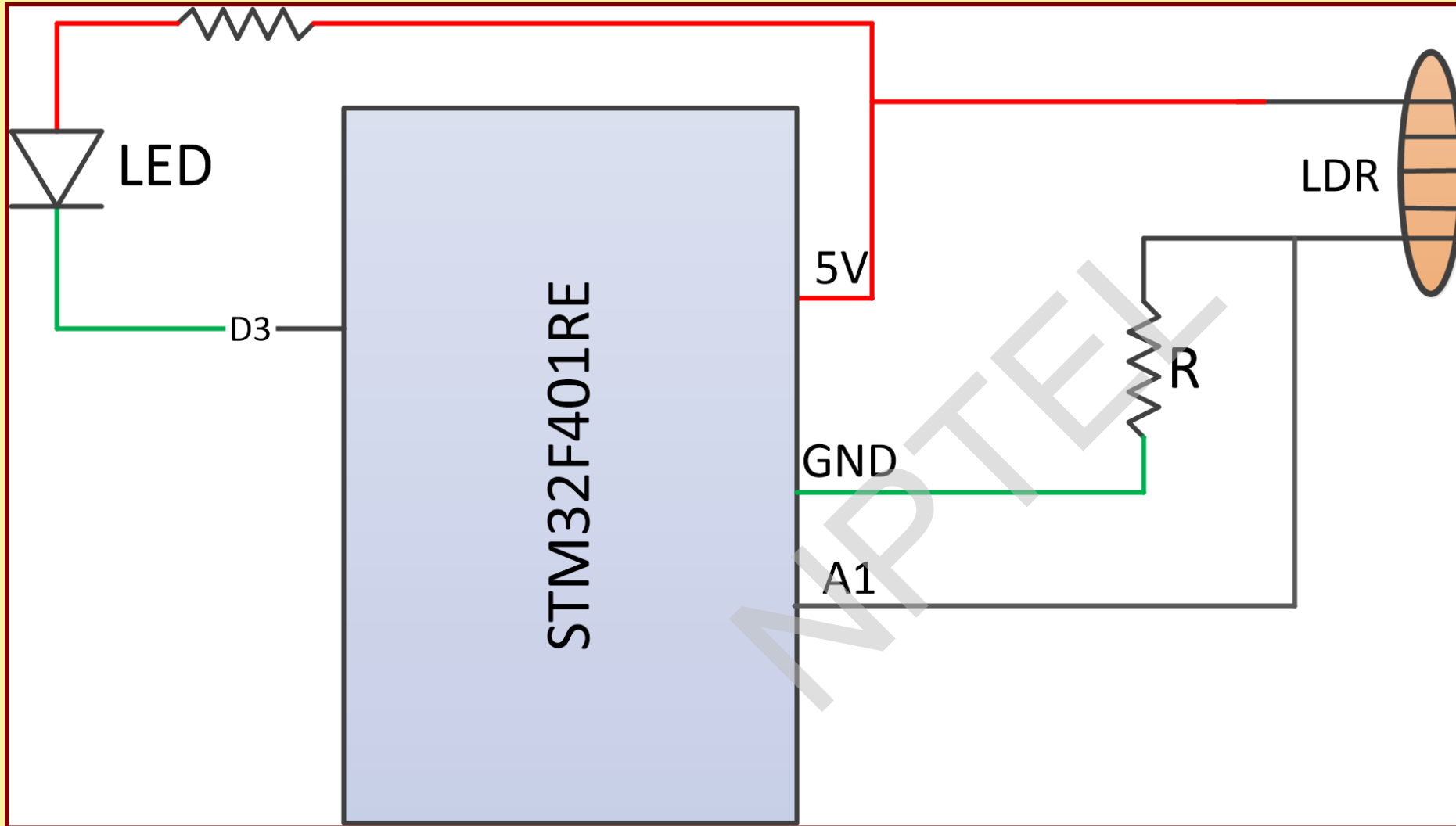


The Experiment



- Create a “*Room Light Controller*” using LDR. Sense the ambient light and adjust the intensity of the room light accordingly.
 - For demonstration, we use a LDR circuit connected to the analog input pin A1 of the Arduino interface.
 - Instead of a bulb (in a room), we use a LED that is connected to the PWM digital output pin D3.
 - The duty cycle of the PWM signal is varied depending on the level of light.
 - In the experiment, we define three levels with corresponding duty cycles 1.0 (fully on), 0.8 (somewhat dim), and 0.0 (off).
 - The LED glows when D3 is at 0.

- How are the threshold values determined?
 - Have been fixed through experimentation.
 - The analog input voltage read from LDR output (0 to 1 range) is multiplied by 5000.
 - The three threshold ranges are fixed at 200, 500 and 850.
 - Less than 200 :: PWM duty cycle = 0.0, and so on.



Connection Diagram

Mbed C Code for STM32

```
#include "mbed.h"
PwmOut LED(D3);
AnalogIn myIn(A1);
int main()
{
    float in1;
    int in2;
    while (1)
    {
        in1 = myIn.read();
        in2 = in1*5000;
```

```
        if (in2>850) {
            LED=1.0;
        }
        else if (in2>500 && in2<850) {
            LED=0.8;
        }
        else if (in2<200) {
            LED=0.0;
        }
    }
}
```



NPTEL ONLINE CERTIFICATION COURSES

**Thank
you!**



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Dr. Kamalika Datta

Department : Computer Science and Engineering

Topic

Lecture 25: Experiment with LDR Light Sensor (Part 2)

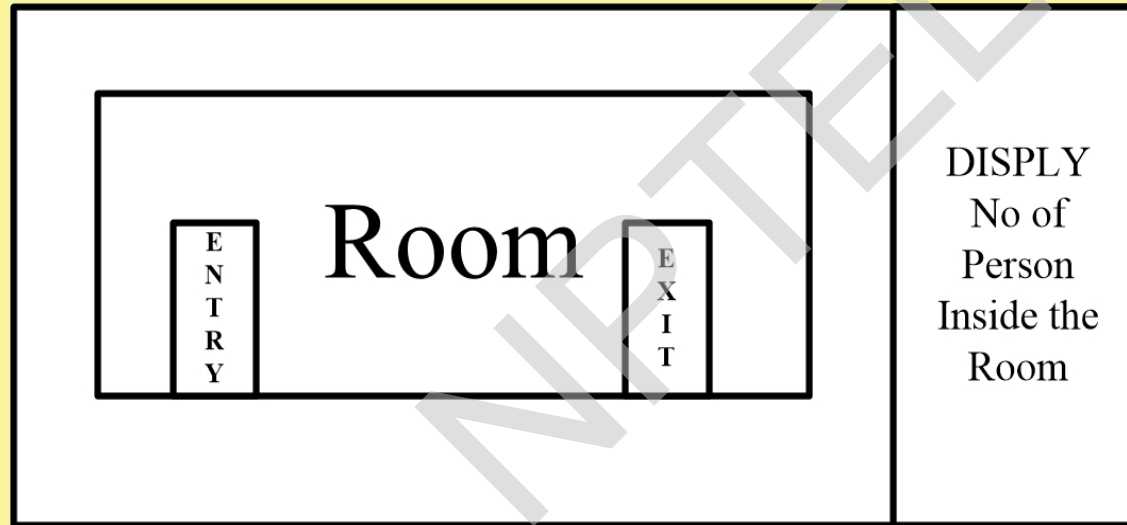
CONCEPTS COVERED

- ☐ Experiment of object counting using LDR light sensor
- ☐ Demonstration



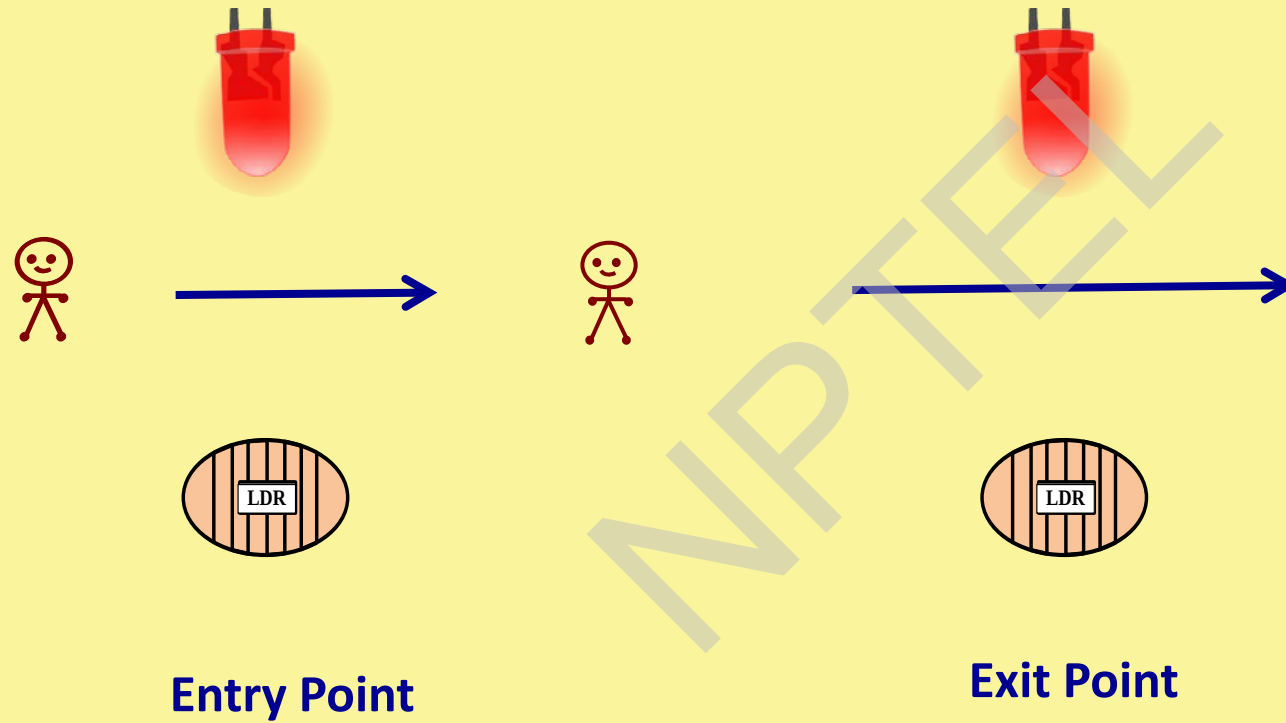
The Experiment

- Design a counter (using LDR and LED's), which will display the number of persons present in a room on a 7-segment LED display



- [Hint: Use two LDRs to detect direction of motion i.e., whether a person is entering or leaving]

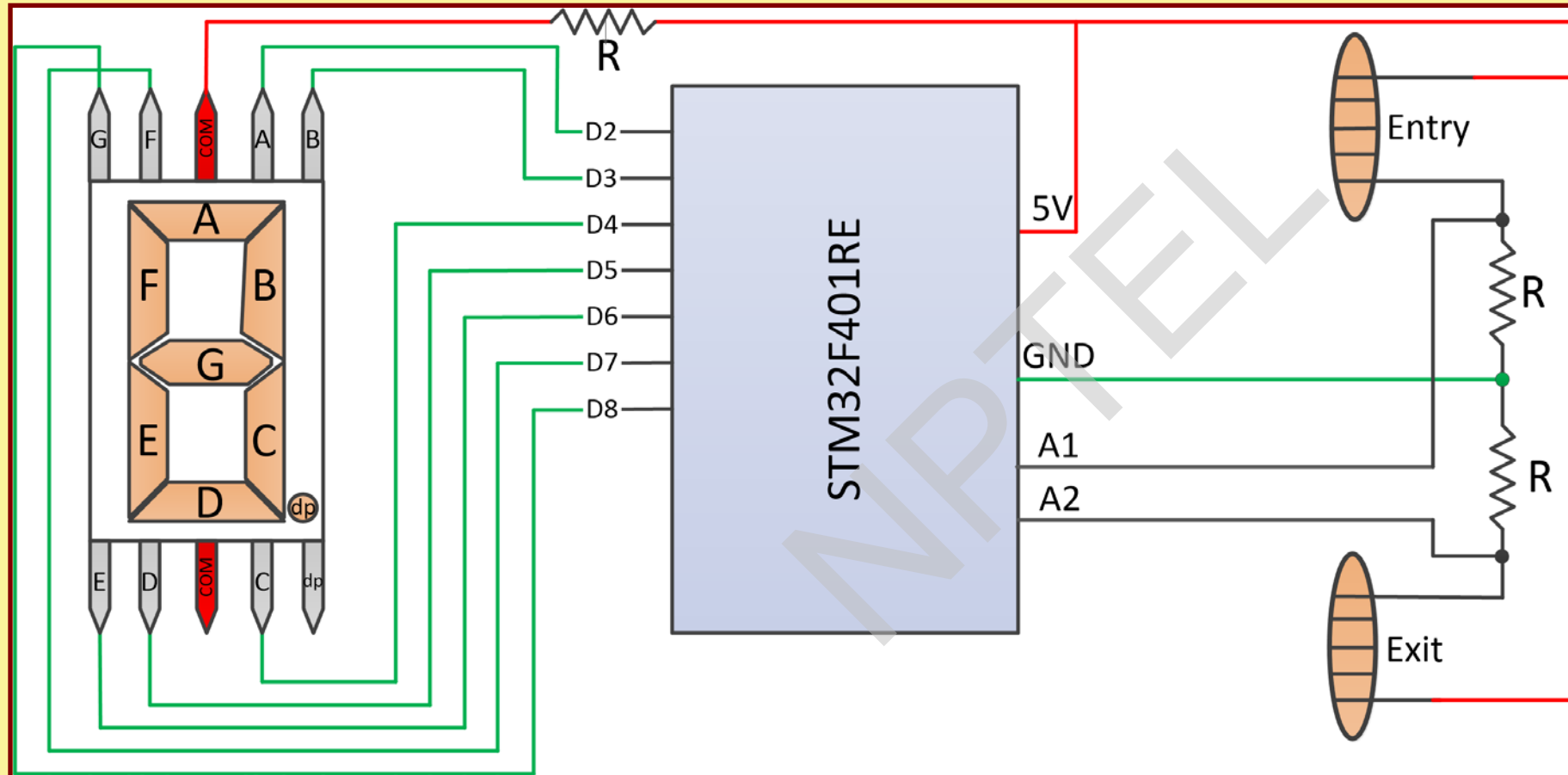
The Arrangement



Basic Concept

- a) Initialize counter to 0.
- b) If the light intensity read by LDR from entry point decreases, increment the counter.
- c) If the light intensity read by LDR from exit point decreases, decrement the counter.

Connection Diagram



The Mbed C Code

```
#include "mbed.h"
#include "string.h"
DigitalOut A(D2);
DigitalOut B(D3);
DigitalOut C(D4);
DigitalOut D(D5);
DigitalOut E(D6);
DigitalOut F(D7);
DigitalOut G(D8);
AnalogIn Entry(A1);
AnalogIn Exit(A2);
```

```
void Display(int disp){
    switch(disp)
    {
        case 0: A=0;B=0;C=0;D=0;E=0;F=0;G=1; break;
        case 1: A=1;B=0;C=0;D=1;E=1;F=1;G=1; break;
        case 2: A=0;B=0;C=1;D=0;E=0;F=1;G=0; break;
        case 3: A=0;B=0;C=0;D=0;E=1;F=1;G=0; break;
        case 4: A=1;B=0;C=0;D=1;E=1;F=0;G=0; break;
        case 5: A=0;B=1;C=0;D=0;E=1;F=0;G=0; break;
        case 6: A=0;B=1;C=0;D=0;E=0;F=0;G=0; break;
        case 7: A=0;B=0;C=0;D=1;E=1;F=1;G=1; break;
        case 8: A=0;B=0;C=0;D=0;E=0;F=0;G=0; break;
        case 9: A=0;B=0;C=0;D=0;E=1;F=0;G=0; break;
    }
}
```

```

int main(){
    float en, ex;
    int per=0;
    while(1) {
        en = Entry.read();
        ex = Exit.read();
        en=en*5000;
        ex=ex*5000;
        if (en<200)
        {
            per=per+1;
            Display(per);
            wait(1);
        }
    }
}

```

```

else if(ex<200) {
    per=per-1; Display(per);
    wait(1);
}
else if (per<=0 || per>9) {
    per=0; Display(per);
    wait(1);
}
else
{
    Display(per);
    wait(1);
}
}
}

```



NPTEL ONLINE CERTIFICATION COURSES

Thank
you!



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Dr. Kamalika Datta

Department : Computer Science and Engineering

Topic

Lecture 28: Experiment with Speaker

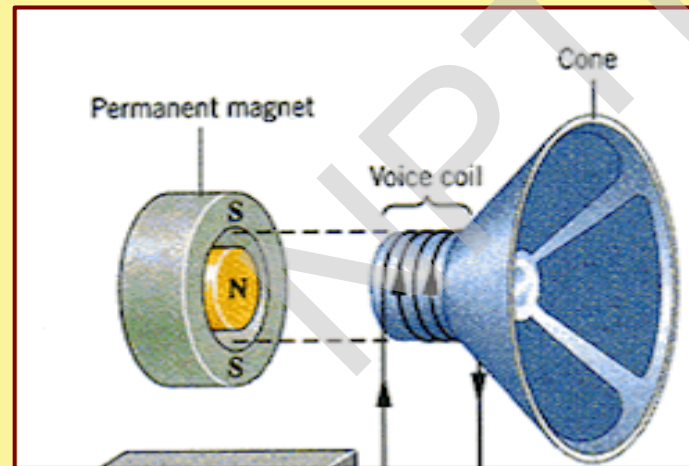
CONCEPTS COVERED

- ☐ Speaker interface circuit
- ☐ Experiments using speaker
- ☐ Demonstration



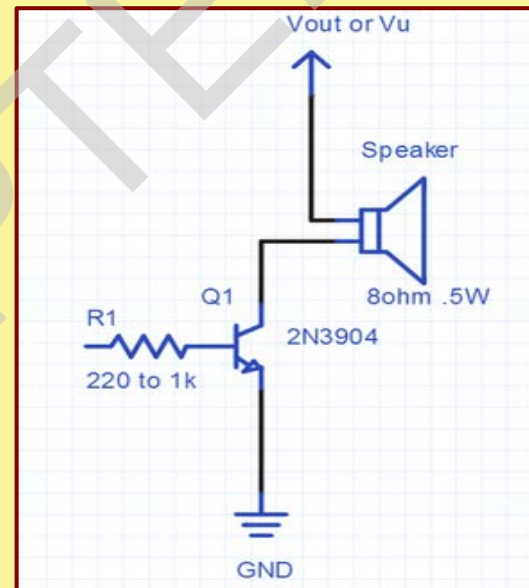
How does a speaker work?

- Speakers work by converting electrical energy to mechanical energy (motion).
 - The mechanical energy compresses air and converts the motion into sound energy or sound pressure level.

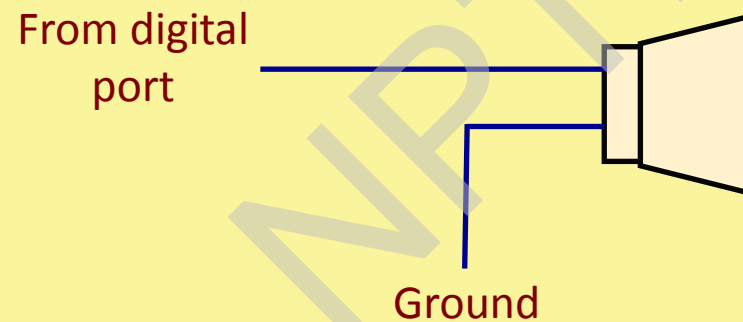


How to Interface a Speaker?

- Any waveform in the audio frequency range from an output port can drive the speaker.
 - May require an amplifier circuit to generate adequate power for the electromagnet.
- A simple transistor based amplifier circuit:



- Small speakers (e.g. 8 ohms or less) can also be driven directly from the port lines of a microcontroller.
 - Of course, sound output will be low.



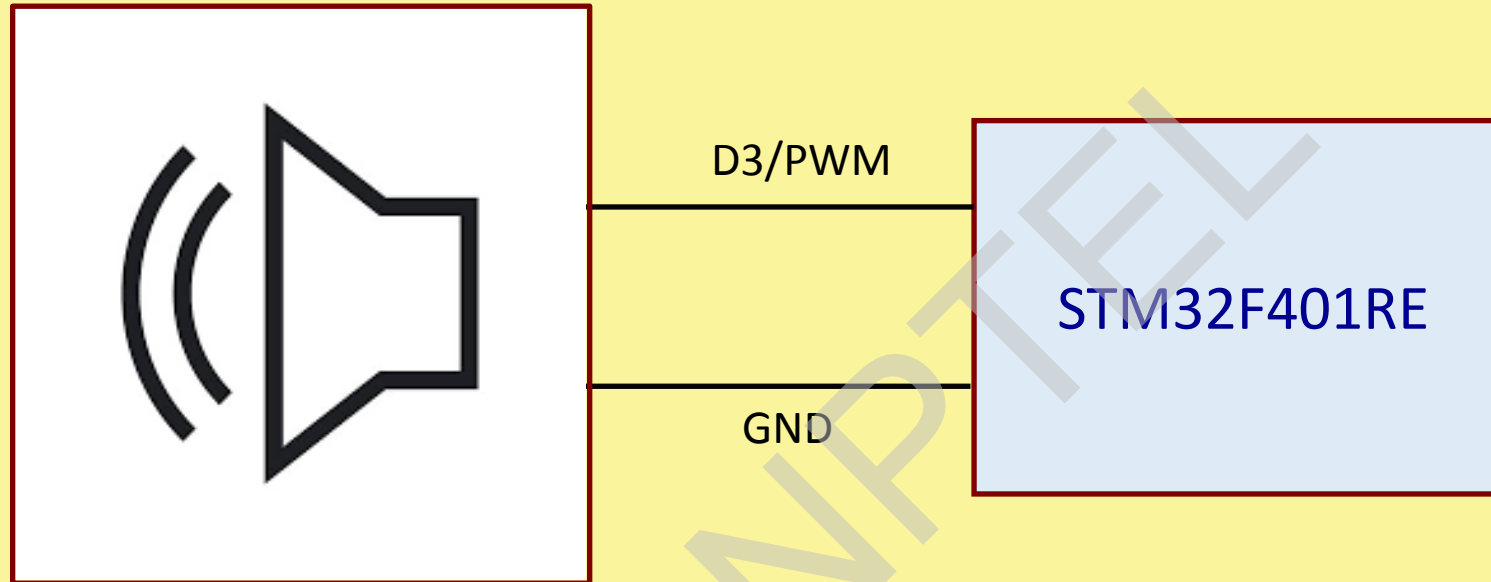
Experiment 1

- Produce a fixed tone on the speaker by generating a square wave of a particular frequency, and use it to drive the speaker.
 - Audible sound is in the frequency range 20 Hz to 20 KHz.
 - In this experiment, we generate a tone with frequency 250 Hz.
 - We can use a pulse width modulated (PWM) port to generate the speaker control signal, by specifying:
 - a) The time period of the generated waveform (i.e. $1 / \text{frequency}$)
 - b) The duty cycle of the generated waveform :: $T_{\text{ON}} / (T_{\text{ON}} + T_{\text{OFF}})$

The PwmOut API :: Relevant Member Functions

- **PwmOut (PinName pin)**
 - Create a PwmOut connected to the specified pin.
- **void write (float value)**
 - Set the output duty cycle, specified as a percentage.
- **void period (float seconds)**
 - Set the PWM period, specified in seconds.

Connection Diagram – STM32



Mbed program for Speaker (beep)

```
#include "mbed.h"
DigitalOut speaker(D3);
int main() {
    while(1) {
        speaker = 0;
        wait (0.002);           // Wait 2 msec
        speaker = 1;
        wait (0.002);           // Wait 2 msec
    }
}
```

This program does not use PWM. It simply outputs 0 and 1 alternately, with time period 4 msec, i.e. frequency of 250 Hz.

Experiment 2

- Generate a two-frequency tone on the speaker, alternating between 333 Hz and 455 Hz, playing each frequency tone for 0.5 second.
 - Play 333 Hz tone for 0.5 second (time period = 3.0 msec).
 - Play 455 Hz tone for 0.5 second (time period = 2.2 msec).
 - Repeat the process in a loop.

Mbed program for Speaker (Ambulance Sound)

```
#include "mbed.h"
PwmOut mypwm(D3);
int main() {
    mypwm.period_us(3000);
    mypwm.pulsewidth_us(1550);
    while(1) {
        mypwm.period_us(3000);
        mypwm.pulsewidth_us(1550);
        wait(0.5); // Play tone-1 for 0.5 sec
        mypwm.period_us(2200);
        mypwm.pulsewidth_us(100);
        wait(0.5); // Play tone-2 for 0.5 sec
    }
}
```

Experiment 3

- Generate the 12 musical notes of an octave on the speaker.
 - Start with the base frequency of 300 Hz.
 - The frequency of the next note is generated by multiplying with 12th root of 2 (i.e. 1.0595).
 - After completing a cycle, it will repeat.

```

#include "mbed.h"
PwmOut speaker(D3);
int main() {
    float freq, period;
    int i;
    speaker.write (0.5);           // Set duty cycle to 0.5
    while (1) {
        freq = 300.0;             // Start with 300 Hz
        for (i=0; i<12; i++) {
            period = 1.0 / freq;
            speaker.period (period);
            wait (0.5);
            freq = freq * 1.0595; // Multiply by 12th root of 2
        }

        wait (1);}
}

```

Mbed program for Speaker (Sa Re Ga Ma)

Experiment 4

- Here the successive notes of a musical piece is played in a repetitive fashion.

```
#include "mbed.h"
```

```
PwmOut speaker(D3);
```

```
int main() {
```

```
    float freq, period;
```

```
    int i;
```

```
    speaker.write (0.5);
```

```
    // Set duty cycle to 0.5
```

```
    while (1) {
```

```
        freq = 262.0;
```

```
        period = 1.0 / freq;
```

```
        speaker.period (period);
```

```
        wait (0.5);
```

```
        freq = 262.0;
```

```
        period = 1.0 / freq;
```

```
        speaker.period (period);
```

```
        wait (0.5);
```

```
        freq = 294.0;
```

```
        period = 1.0 / freq;
```

```
        speaker.period (period);
```

```
        wait (0.5);
```

```
        freq = 262.0;
```

```
        period = 1.0 / freq;
```

```
        speaker.period (period);
```

```
        wait (0.5);
```

```
        freq = 349.0;
```

```
        period = 1.0 / freq;
```

```
        speaker.period (period);
```

```
        wait (0.5);
```

```
        freq = 330.0;
```

```
        period = 1.0 / freq;
```

```
        speaker.period (period);
```

```
        wait (0.5);
```

```
        wait (0.5);
```

```
    }
```

```
}
```

Mbed program for Speaker (Musical Tone)



NPTEL ONLINE CERTIFICATION COURSES

Thank
you!



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Dr. Kamalika Datta

Department : Computer Science and Engineering

Topic

Lecture 29: Experiments with Microphone

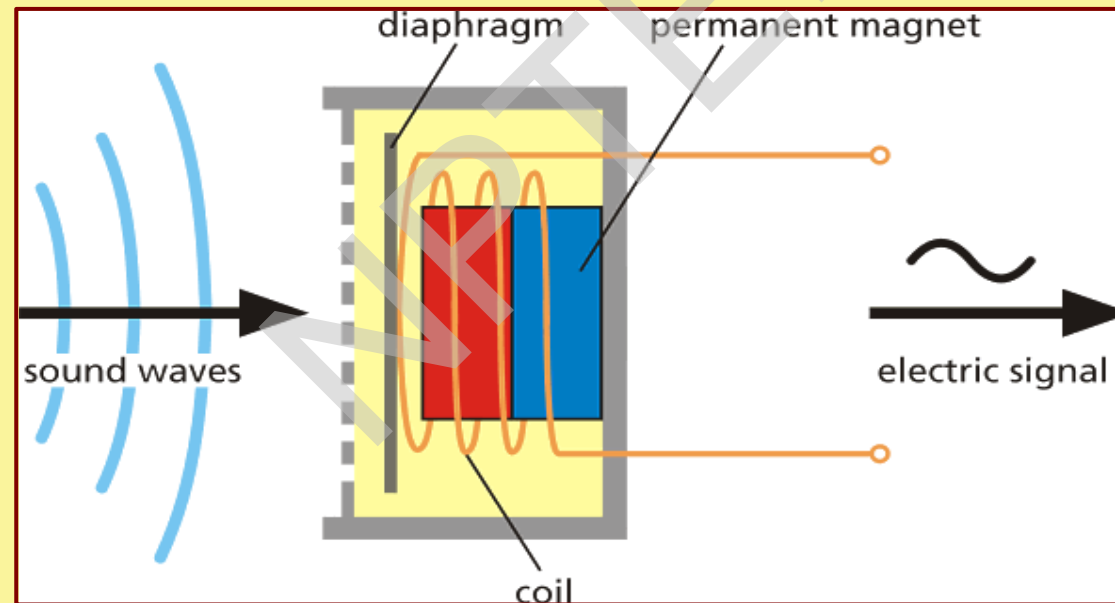
CONCEPTS COVERED

- ☐ Microphone interfacing
- ☐ Experiments using microphone
- ☐ Demonstration

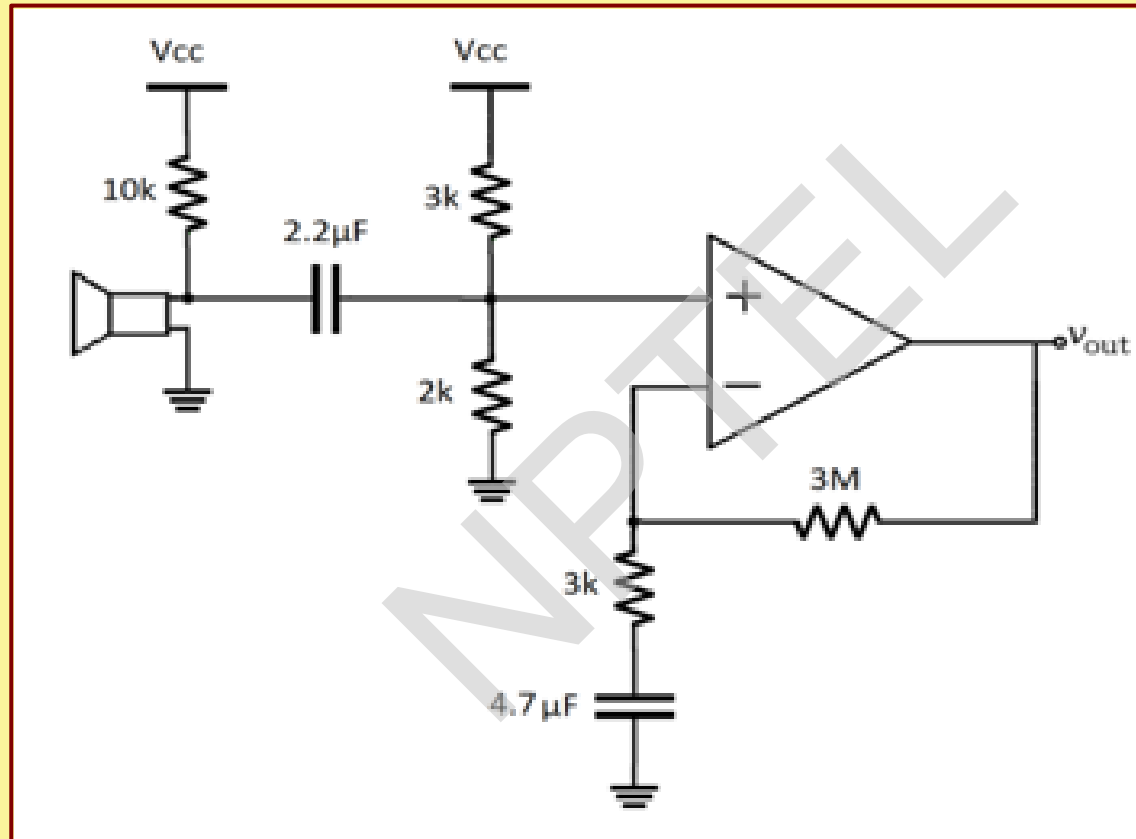


What is a Microphone?

- A microphone is a type of transducer, that converts acoustical energy (sound waves) into electrical energy (the audio signal).

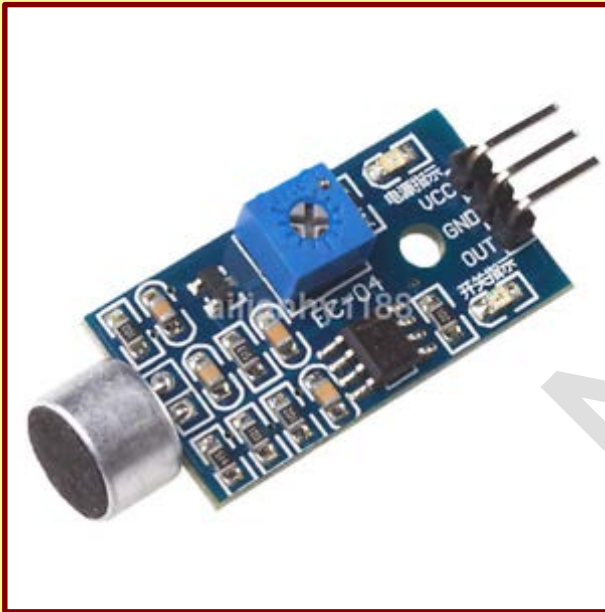


Typical Microphone Interfacing Circuit



Microphone Module with Interface

- For ease of interfacing, microphone with driver circuit is available as a module.

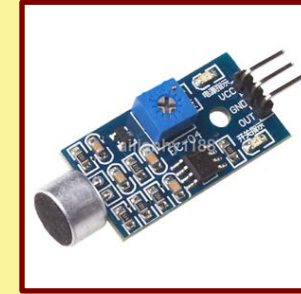
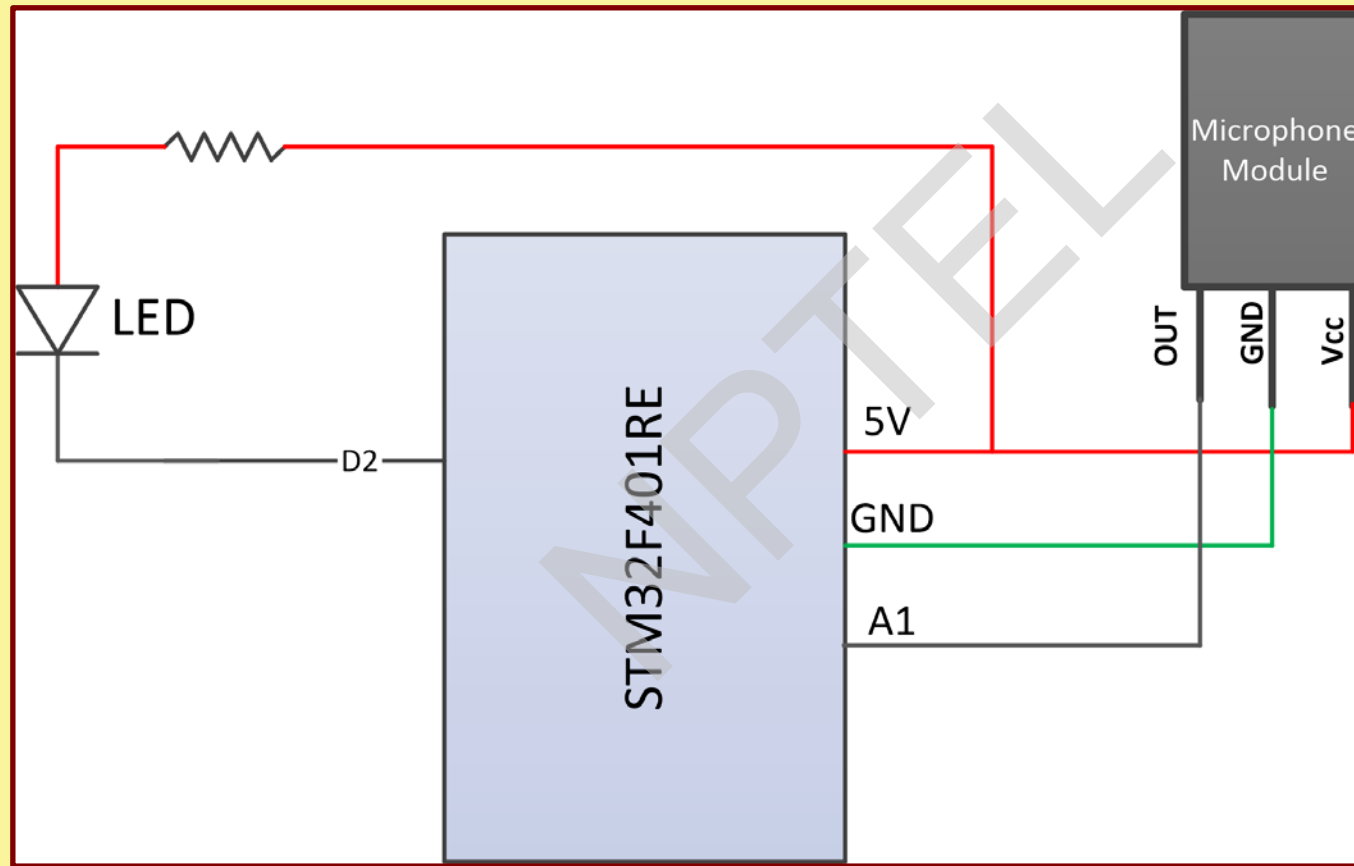


- Can work with power supply of 3.3V to 5.0V.
- Digital output (0 or 1), depending on whether sound is detected or not.
- Can be directly interfaced with the microcontroller.

Experiment 1

- Implement a clap switch, where a LED can be turned ON using the sound from a clap.
 - The microphone circuit is interfaced to the *AnalogIn* pin *A1*.
 - The *AnalogIn* object converts the input voltage to floating-point numbers from 0.0 to 1.0.
 - Through experimentation, we can determine the threshold value, which would depend on the noise of the environment.
 - The microphone module used, however, generates a 2-level digital output.
 - A LED is connected to the *DigitalOut* pin *D2*, where the LED glows if the *D2* pin is at 0.
 - The LED turns on for 3 seconds, and then turns off.

Connection Diagram – STM32



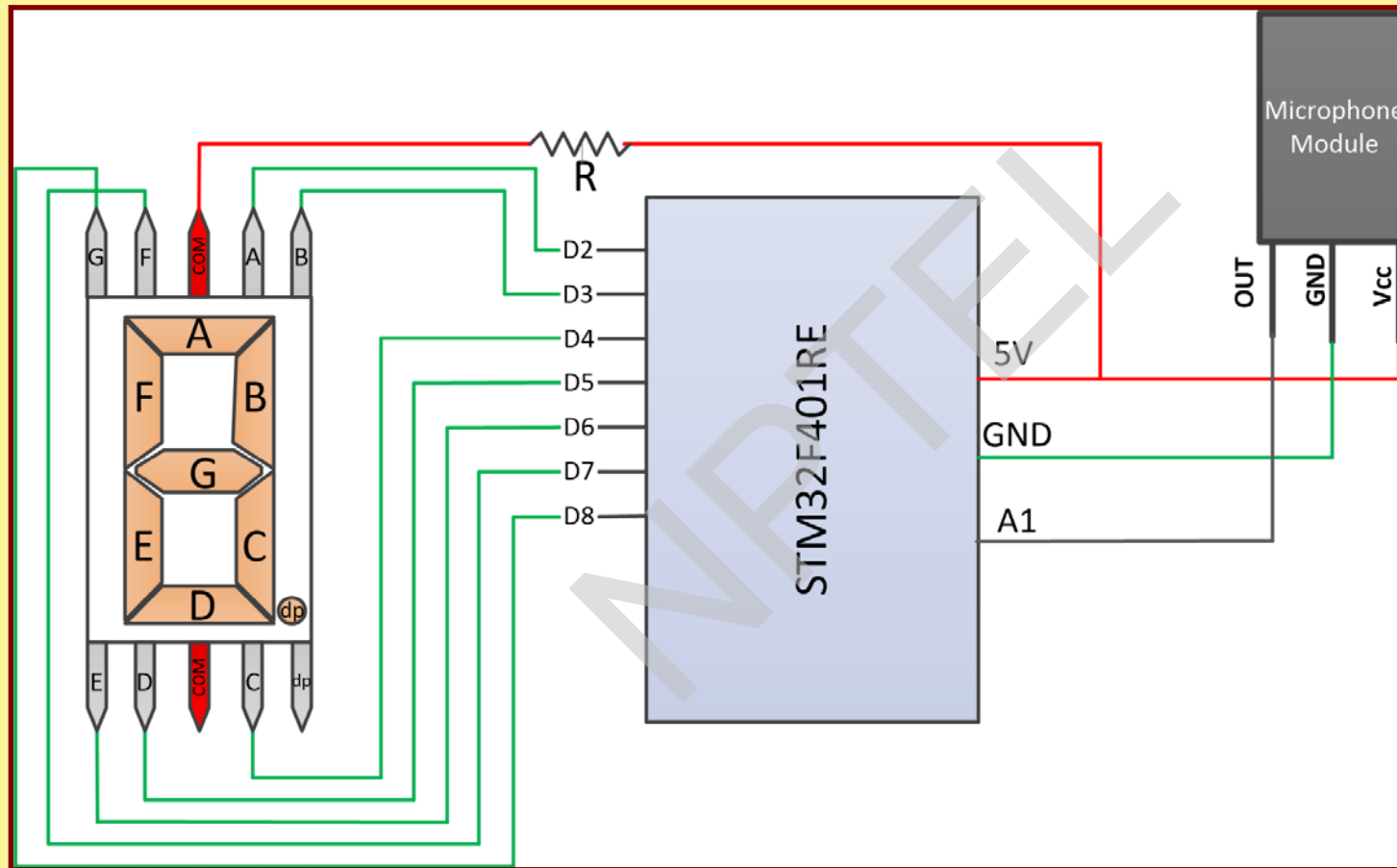
Mbed program for Microphone

```
#include "mbed.h"
AnalogIn analog_value(A1);
DigitalOut led(D2);
int main() {
    float meas;
    while(1) {
        led = 1;
        meas = analog_value.read();
        if (meas>=0.7){
            led = 0;
            wait(3);
            led = 1;
        }
    }
}
```

Experiment 2

- Interface a microphone and a 7-segment LED display unit to the microcontroller, and display the number of claps on the 7-segment display.

Connection Diagram – STM32



Mbed program for Microphone (Clap Count)

```
#include "mbed.h"
DigitalOut A(D2);
DigitalOut B(D3);
DigitalOut C(D4);
DigitalOut D(D5);
DigitalOut E(D6);
DigitalOut F(D7);
DigitalOut G(D8);
AnalogIn val(A1);
```

```
void Display(int disp) {
    switch(disp)
    {
        case 0: A=0;B=0;C=0;D=0;E=0;F=0;G=1; break;
        case 1: A=1;B=0;C=0;D=1;E=1;F=1;G=1; break;
        case 2: A=0;B=0;C=1;D=0;E=0;F=1;G=0; break;
        case 3: A=0;B=0;C=0;D=0;E=1;F=1;G=0; break;
        case 4: A=1;B=0;C=0;D=1;E=1;F=0;G=0; break;
        case 5: A=0;B=1;C=0;D=0;E=1;F=0;G=0; break;
        case 6: A=0;B=1;C=0;D=0;E=0;F=0;G=0; break;
        case 7: A=0;B=0;C=0;D=1;E=1;F=1;G=1; break;
        case 8: A=0;B=0;C=0;D=0;E=0;F=0;G=0; break;
        case 9: A=0;B=0;C=0;D=0;E=1;F=0;G=0; break;
    }
}
```

```
int num=0;
int main()
{
    float meas;
    while(1) {
        meas = val.read();
        if (meas >= 0.7) {
            num++;
            wait(0.1);
        }
        if (num > 9) {
            num=0;
        }
        Display(num);
    }
}
```



NPTEL ONLINE CERTIFICATION COURSES

Thank
you!