# Morse Code Decoder

*Microprocessor & Microcontroller*

*Sessional*

**CSE 316**

Student ID:

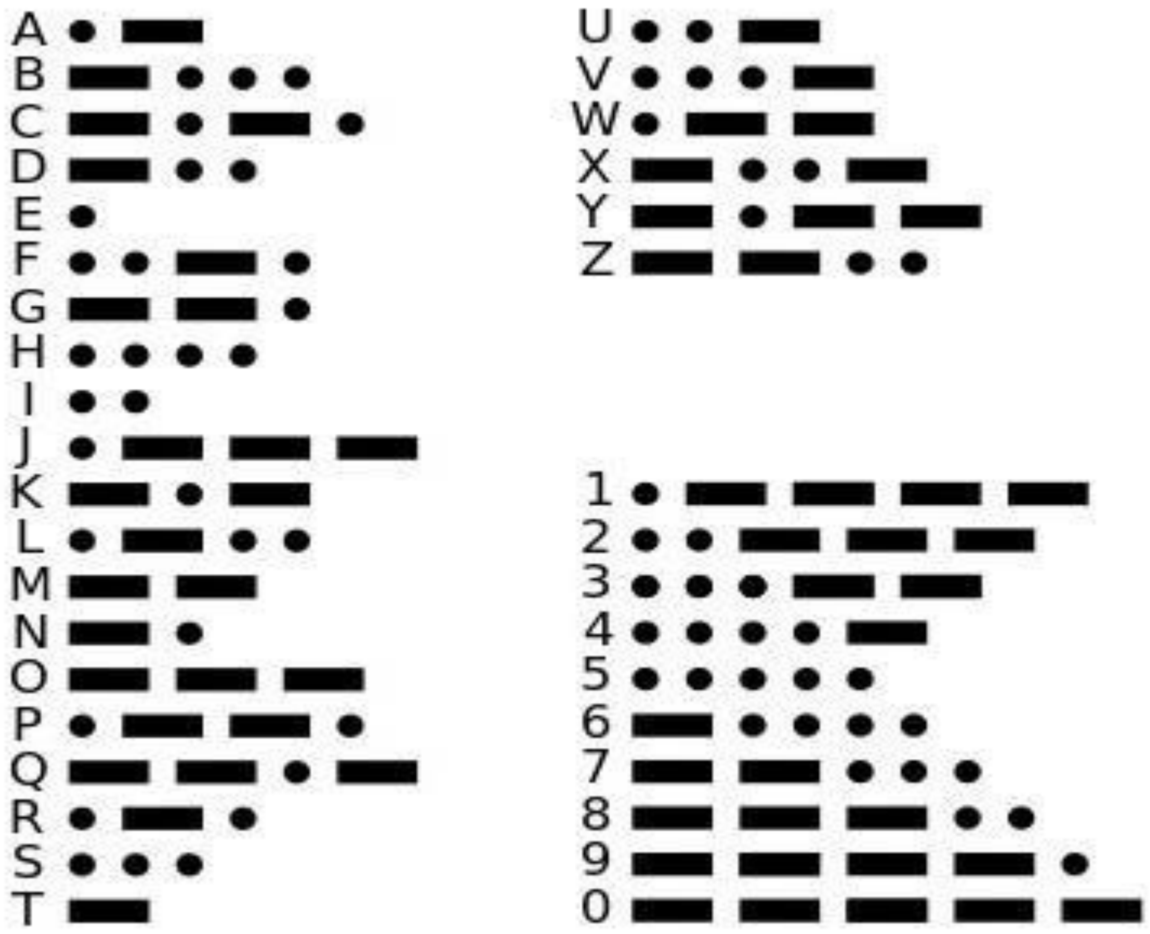
Md. Rashid Abid (**1205016**)

Muhim Muktadir Khan (**1205026**)

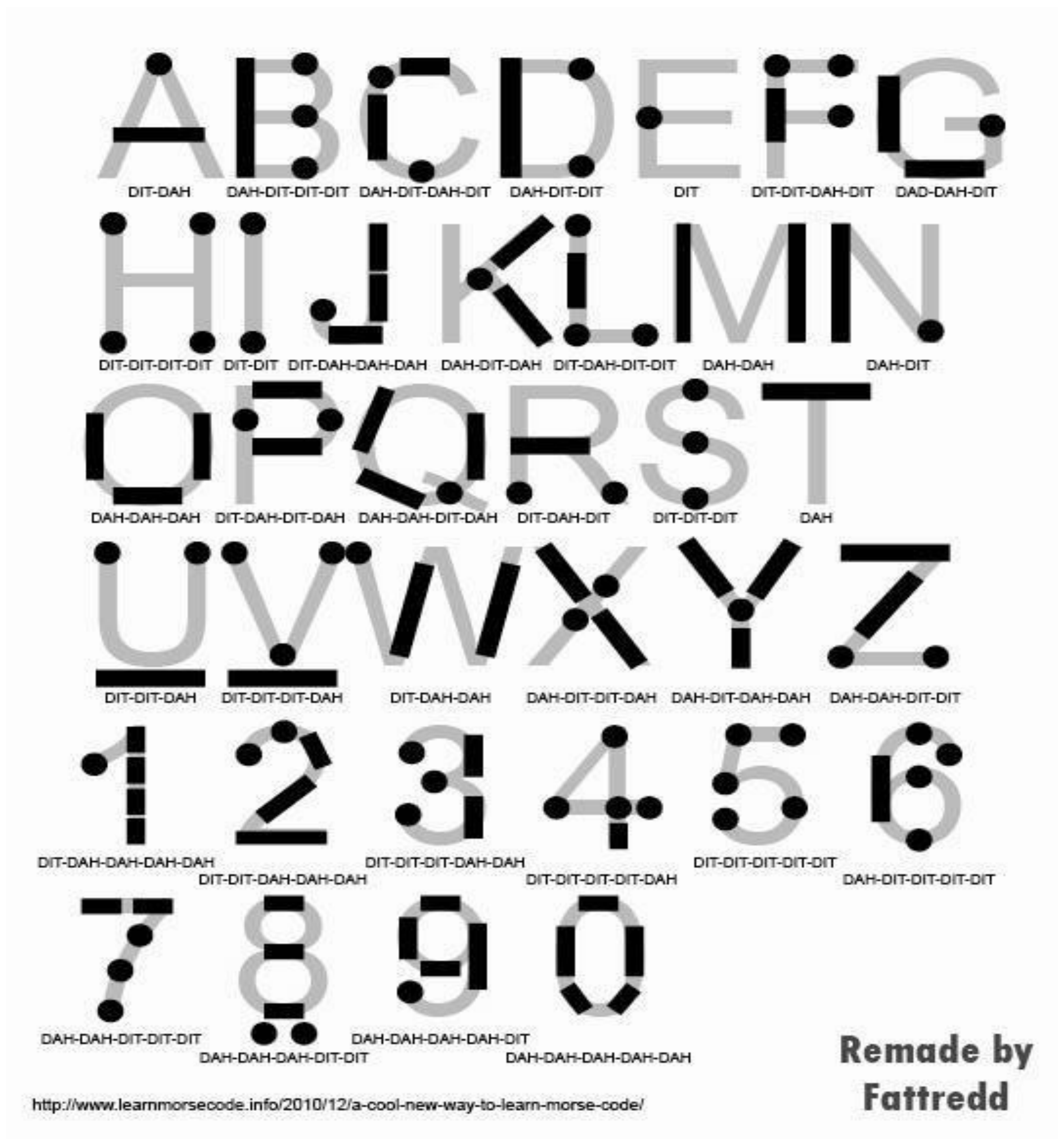
**<u>Project Name</u>**: Morse Code Decoder (**1205016**, **1205026**)

**<u>Brief Description about Morse Code</u>**:

Morse code is maybe the oldest and simplest way for transmitting text data in encoded form. Here, each text letter is considered as a combination of “dots-and-dashes”.



| | |
|---|---|
| A | ● ▬ |
| B | ▬ ● ● ● |
| C | ▬ ● ▬ ● |
| D | ▬ ● ● |
| E | ● |
| F | ● ● ▬ ● |
| G | ▬ ▬ ● |
| H | ● ● ● ● |
| I | ● ● |
| J | ● ▬ ▬ ▬ |
| K | ▬ ● ▬ |
| L | ● ▬ ● ● |
| M | ▬ ▬ |
| N | ▬ ● |
| O | ▬ ▬ ▬ |
| P | ● ▬ ▬ ● |
| Q | ▬ ▬ ● ▬ |
| R | ● ▬ ● |
| S | ● ● ● |
| T | ▬ |
| U | ● ● ▬ |
| V | ● ● ● ▬ |
| W | ● ▬ ▬ |
| X | ▬ ● ● ▬ |
| Y | ▬ ● ▬ ▬ |
| Z | ▬ ▬ ● ● |
| 1 | ● ▬ ▬ ▬ ▬ |
| 2 | ● ● ▬ ▬ ▬ |
| 3 | ● ● ● ▬ ▬ |
| 4 | ● ● ● ● ▬ |
| 5 | ● ● ● ● ● |
| 6 | ▬ ● ● ● ● |
| 7 | ▬ ▬ ● ● ● |
| 8 | ▬ ▬ ▬ ● ● |
| 9 | ▬ ▬ ▬ ▬ ● |
| 0 | ▬ ▬ ▬ ▬ ▬ |



This concept of dash and dot can be implemented with on-off tones, lights, clicks etc. A skilled listener will be able to decode the text message from the Morse-Code.

## **Morse Code Implementing Rules**:

The duration of a dash is three times the duration of a dot. Each dot or dash is followed by a short silence, equal to the dot duration. The letters of a word are separated by a space equal to three dots (one dash), and the words are separated by a space equal to seven dots. The dot duration is the basic unit of time measurement in code transmission. To increase the speed of the communication, the code was designed so that the length of each character in Morse varies approximately inversely to its frequency of occurrence in English. Thus the most common letter in English, the letter "E", has the shortest code, a single dot.

Details: https://en.wikipedia.org/wiki/Morse_code

## **Morse Code Decoder**:

Previously, Morse codes were decoded through a mechanical device and it was widely used in military particularly. To demonstrate how the mechanical device worked, here are two video links:

1) https://www.youtube.com/watch?v=xsDk5_bktFo
2) https://www.youtube.com/watch?v=JT9zM_-2S6g

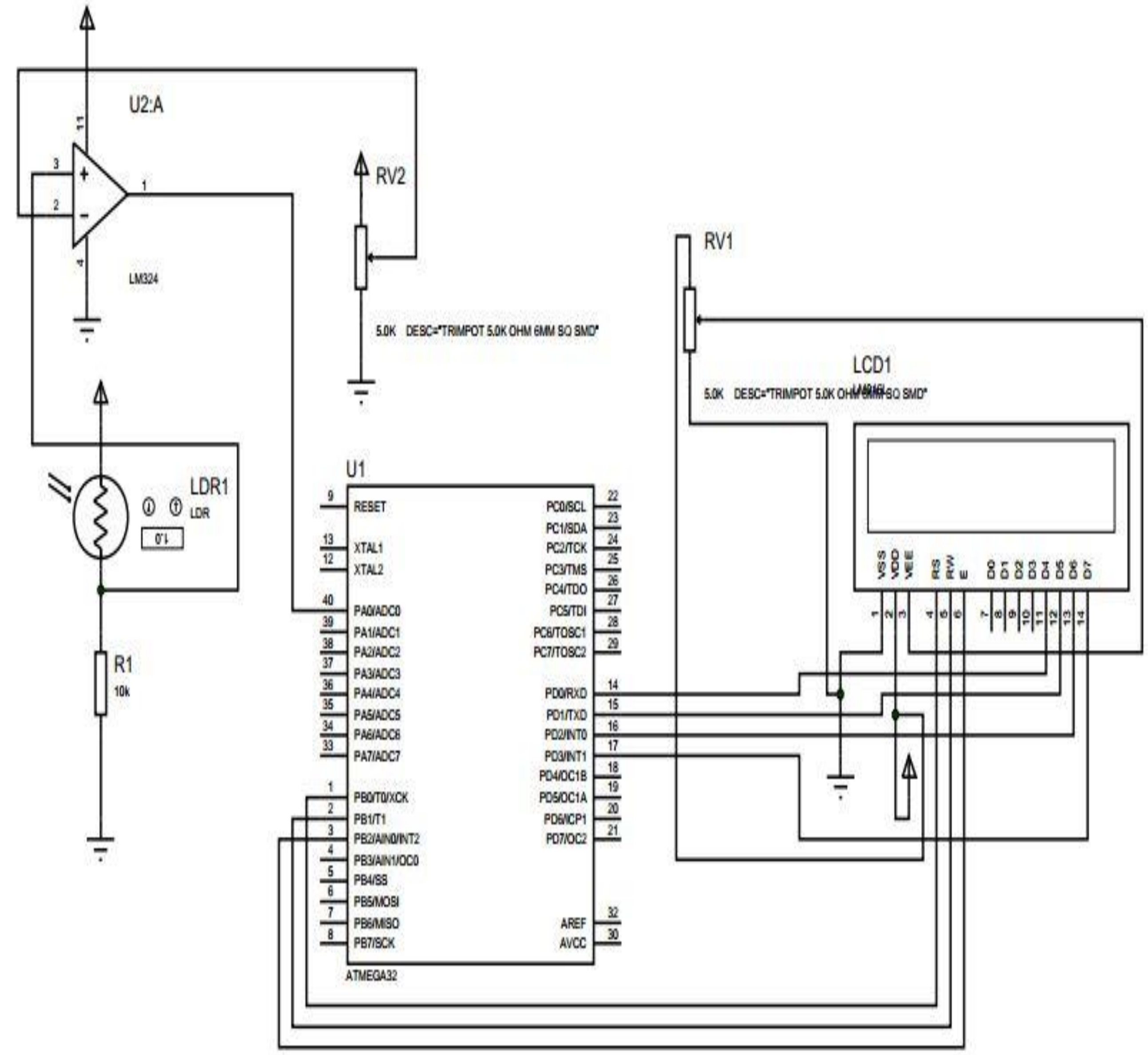
## **Problem Specification**:

In this project, we implemented the Morse code decoder not as a mechanical device, but with the help of a microcontroller. Now it can be integrated further with other digital devices if necessary.

**<u>Solution Overview</u>**:

- We implemented the “dots-and dashes” by giving laser light rays as input. A LDR(Light Dependent Resistor) reacts to the light ray inputs accordingly. As we already know, the duration of dashes is three times the duration of dots. So, if we provide light to the LDR for a certain duration and call it “dot”, then we have to provide light for a longer duration and call it “dash”. We did accordingly here.
- The LDR detects the duration of light and understand whether it is a dot or a dash. Then the respective dot-dash code format tells us the text character. This text character is shown in a text display.
- Thus, a series of dots and dashes gets decoded by the microcontroller and is shown as a text message in the text display.
- We also implemented a separate system for giving ‘dot’ and ‘dash’ with the help of two push buttons. These push buttons make it easier to put appropriate duration of lights.
- In case it doesn’t work perfectly, we also **built** a separate **ANDROID APP** where if we write down a text, it converts the text to the corresponding dot-dash format of lights matching the time-requirements according to our project.

## Circuit Diagram:



## Source Code (Morse Code Decoder Unit):

```
/*
 * Morse Code.cpp
 *
 * Created: 5/18/2016 12:21:35 AM
 * Author : you
 */

#include <avr/io.h>
#include "lcd.h"

#define ditTime 50
#define dahTime 100
#define darkTime 200

//enum code {dit=0,dah=1,dark=2};

#define dit 1
#define dah 2
#define dark 0


int morseCodeMap [37];
int curX,curY;

void mapInit()
{
	morseCodeMap[0]=12;
	morseCodeMap[1]=2111;
	morseCodeMap[2]=2121;
	morseCodeMap[3]=211;
	morseCodeMap[4]=1;
	morseCodeMap[5]=1121;
	morseCodeMap[6]=221;
	morseCodeMap[7]=1111;
	morseCodeMap[8]=11;
	morseCodeMap[9]=1222;
	morseCodeMap[10]=212;
	morseCodeMap[11]=1211;
```

```
        morseCodeMap[12]=22;
        morseCodeMap[13]=21;
        morseCodeMap[14]=222;
        morseCodeMap[15]=1221;
        morseCodeMap[16]=2212;
        morseCodeMap[17]=121;
        morseCodeMap[18]=111;
        morseCodeMap[19]=2;
        morseCodeMap[20]=112;
        morseCodeMap[21]=1112;
        morseCodeMap[22]=122;
        morseCodeMap[23]=2112;
        morseCodeMap[24]=2122;
        morseCodeMap[25]=2211;
        morseCodeMap[26]=12222;
        morseCodeMap[27]=11222;
        morseCodeMap[28]=11122;
        morseCodeMap[29]=11112;
        morseCodeMap[30]=11111;
        morseCodeMap[31]=21111;
        morseCodeMap[32]=22111;
        morseCodeMap[33]=22211;
        morseCodeMap[34]=22221;
        morseCodeMap[35]=22222;
        morseCodeMap[36]=32; //space

}

void printLCD(char c)
{
        //PORTB=(temp%3)+1;
        curX++;
        if(curX>16)
        {
                curY++;
        }
        if (curY>2 && curX>16)
        {
                curY=1;
                curX=1;
        }
        if(curX>16) curX=1;


        GotoMrLCDsLocation(curX,curY);
        Send_A_Character(c);
        //Send_A_String("x");
        //_delay_ms(50);


}

void decodeMorse(int key,int len)
{
        //printLCD(65);
        int i,sendChar;
        for(i=0;i<36;i++)
        {
              if(morseCodeMap[i]==key)
              {
                     if(i<26) sendChar=65+i;
                     else if(i<35) sendChar= 23+i;
                     else sendChar = 48;

                     printLCD(sendChar);
                     break;
              }
       }

}




int main(void)
{
    DDRB = 0b11111111;
       DDRD = 0b00000000;
       PORTB = 0b00000000;

       TCCR1B |= 1<<CS10;
       PORTD = 1;

       mapInit();

       int count = 0;
       int darkCount=0;
       int key=0;
       int keyLength=0;

       InitializeMrLCD();
       curX=1;
       curY=1;



    while (1)
    {
              if(TCNT1 >=10000)
              {
                     TCNT1 = 0;
                     //count++;

                     //char n = PIND;

                     if(bit_is_clear(PIND,0)==0)
                     {
                            //printLCD('B');
                            count++;
                            darkCount=0;
                     }
                     else
                     {
                            darkCount++;
```

```
                        if(keyLength>5)
                        {
                                //printLCD('G');
                                keyLength=0;
                                key=0;
                        }

                        if(darkCount>=darkTime)
                        {
                                //printLCD('D');
                                decodeMorse(key,keyLength);
                                keyLength=0;
                                key=0;
                                darkCount= 0;

                        }

                        else if (count>=ditTime && count<dahTime)
                        {
                                //printLCD('I');
                                key=key*10+dit;
                                keyLength++;
                        }

                        else if(count>=dahTime)
                        {
                                //printLCD('A');
                                key=key*10+dah;
                                keyLength++;

                        }
                        //else printLCD('N');

                        count=0;

                }

        }

  }
}
```

## Source Code (Pushbutton Unit):

```
/*
 * push button.cpp
 *
 * Created: 06-Jun-16 4:16:37 AM
 * Author : you
 */

#include<avr/io.h>
#include<util/delay.h>
#define F_CPU 1000000
#include<avr/interrupt.h>
```

```
ISR(INT0_vect)
{
	PORTB=0b00000011;
	_delay_ms(500);
	PORTB=0b00000000;
}

ISR(INT1_vect)
{
	PORTB=0b00000011;
	_delay_ms(1000);
	PORTB=0b00000000;
}


int main(void)
{

	DDRB = 0b00000011;

	PORTB=0;

	GICR = GICR | (1<<INT0);
	GICR = GICR | (1<<INT1);

	MCUCR = 0b00001111;

	sei();

	while(1)
	{

	}
}
```

## Challenges:

1. **Power:** First we tried to take power from the USB cable connecting with the PC. But, it cannot provide voltage significantly lower than 5.0 volts. As a result, to make sure we get 5.0 volts in the microcontroller, we took power from the external source using **Adapter** and a **7805 Diode**.

2. **LDR Calibration:** To make sure LDR works perfectly, we first calibrated it with **POT** (Potentiometre) and then put exact amount

of resistances in the circuit and avoided POT.

3. **Digital Input:** We had to make sure the voltage provided by LDR maintains a specific range. Voltages below 2.5 Volts are considered are 0 and voltages upper 2.5 Volts are considered as 1. We could have used microcontroller's **ADC** feature, but we chose to implement it in hardware and used an **Op-Amp.** It gives us those binary inputs.

4. **Time Calibration:** We needed to make sure the microcontroller can distinguish 'dot', 'dash' and 'silence' by calculating the amount of time light is applied on or darkness is remained.

5. **LDR Broken Light Inputs:** Sometimes, the LDR was not behaved accordingly and could not receive continuous light inputs, rather break it in pieces. Later, we discovered that the resistances were not calibrated properly.

6. **Push Button Misbehave:** Sometimes, the push buttons give a slight higher amount of light as 'dot' and much higher/sometimes lesser amount of light as 'dash'. For this reason, we don't solely depend on the hardware circuit rather use software technique, the app we built in **Android.**

7. **LCD Problems:** LCD was not moving cursors after a line end. Then we used appropriate header for 16*2 LCD display and could do that accordingly. Also, we faced problems in selecting LCD contrast. Then switching on **'backlight'** integrated with the LCD made it finer.