# COMMUNICATION SYSTEM FOR THE DISABLED USING MORSE CODE

**J Component Project Report for the course**

**CSE2006 Microprocessor and Interfacing**

*by*

**Bhavishya Kumar (20BCE1874)**
**Josiah James (20BCE1344)**
**Ayesha Numa (20BCE1290)**

*Submitted to*

**Dr. Muthulakshmi S**

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
VELLORE INSTITUTE OF TECHNOLOGY
CHENNAI - 600127
*April 2022*

## *Certificate*

This is to certify that the Project work titled “COMMUNICATION FOR THE DISABLED USING MORSE CODE” is being submitted by Bhavishya Kumar (20BCE1874), Josiah James (20BCE1344), Ayesha Numa (20BCE1290) for the course **Microprocessor and Interfacing**, is a record of bonafide work done under my guidance. The contents of this project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University.

**Dr. Muthulakshmi S**

## ABSTRACT

Communication is the most basic requirement of a human being. But unfortunately this basic need is not fulfilled in the life of disabled people. Lack of communication creates various problems for the patient. Speech disabled people can use symbolic languages to express themselves. But this condition is worsened with people having multiple disabilities like severe physical restriction along with verbal disabilities as in case of cerebral palsy, multiple sclerosis etc. To assist such people, Morse code has always proved itself to be a great tool. Morse code is a method of transmitting information as a series of on-off tones, lights, or clicks that can be directly understood by a listener or observer without any special equipment.

In this method, each letter or number is represented by using short (dot) and long (dash) elements. Morse code, like any other language, has its own alphabet.

It is also a great method of communication during an emergency because of its bandwidth efficiency, low transmission power, immunity to interference and difficulty in decoding.

We believe that the ancient methods of communication should not just be left in the books and efforts must be made to keep them alive.

We tend to send the Morse code to Arduino using the keys and see the result as text in the Serial Monitor, type our desired word or text and receive it in Morse code, so that we can send it as light or sound and effectively use Morse code to facilitate communication for the mute, deaf and blind.

# Table of Contents

## List of Figures

# CHAPTER 1

# Introduction

The disabled people face a variety of issues as a result of  lack of communication. People who are unable to communicate verbally can express themselves using symbolic languages. People with multiple impairments, such as cerebral palsy or multiple sclerosis, have a worsening of this disease since they have severe physical restrictions as well as speech limitations. Morse code has always shown to be a valuable tool in assisting such persons. Morse code is a way of communicating information in the form of a succession of on-off tones, lights, or clicks that can be directly comprehended by a listener or observer without the use of any additional equipment.Morse code, like any other language, has its own alphabet and is currently available in both American and International types, and the most commonly used one is international type. Morse code can be transmitted in different ways: originally as electrical pulses along a telegraph wire, but also as an audio tone, a radio signal, light, body language, frequency and more. Imagine the dot as a unit of time, then the dash is three units of time, the distance between the parts of a letter is a unit of time, the distance between two consecutive letters is three units of time, and the distance between the words is seven units of time.

## 1.1 Purpose

The following are the objectives of this project:

1. To send the Morse code to Arduino using the keys and see the result as text in the Serial Monitor.

2. To type our desired word or text and receive it in Morse code, so that we can send it as light or sound.

3. To effectively use Morse code to facilitate communication for the mute,deaf and blind.

## 1.2 Scope

Morse code with an easy-to-operate, single switch input system has been shown to be an excellent communication adaptive device. Because maintaining a stable typing rate is not easy for the disabled, the automatic recognition of Morse code is difficult. Therefore, a suitable adaptive automatic recognition method is needed.

Morse Code is also prevalent in Aviation and Aeronautical fields since radio navigational aids such as VOR's and NDB's still identify in Morse Code. The US Navy and Coast Guard still use signal lamps to communicate via Morse Code.

Another group that's showing it some love is the International Morse Code Preservation Society — a coalition of amateur radio operators with thousands of members around the globe. So while the golden age of dots and dashes may be over, Morse code is still hanging in there. No distress signal required.

And besides all, learning and using Morse code to communicate can be fun and entertaining.

# CHAPTER 2

# Design/Implementation

## 2.1 Introduction

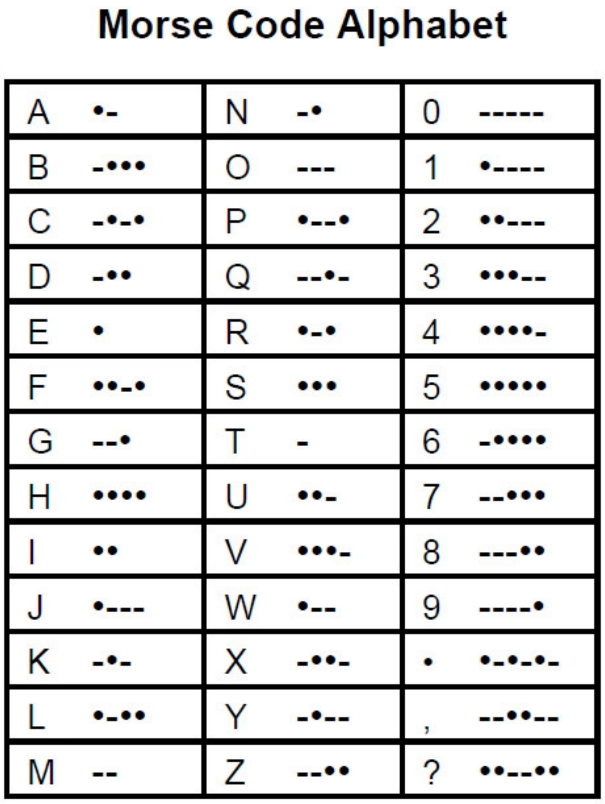
The Morse code encoder takes input using the keyboard in English and converts it to Morse code. The converted morse code can be seen on the serial monitor, heard as beeps using the buzzer and signalled using a blinking LED.

The Morse code decoder takes input using a button, a longer press represents a dash whereas a short one is for a dot. The LED blinks when we feed in the input in morse code. The converted code in English is visible on the Serial monitor.

It can be used as an efficient medium of communication by the deaf,blind and dumb.

Furthermore, we can use it to send and receive encrypted messages.

The codes are uploaded and compiled on the Arduino software. The microcontroller used is the Arduino UNO R3. The breadboard facilitates in making circuit connections.



**Morse Code Alphabet**

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

**Figure 2.1 Morse Code Chart**

## 2.2 Design Approach

**Encoder**

The dot function is used to depict a dot in morse code.The LED and the buzzer are in sync with each other. DigitalWrite allows the led to blink and buzzer to beep in high and then low with unit delay in between for a dot.

```
 void dot()
{
Serial.print(".");
digitalWrite(led, HIGH);
digitalWrite(buz, HIGH);
delay(unit_delay);
digitalWrite(led, LOW);
digitalWrite(buz, LOW);
delay(unit_delay);
}
```

The dash function is used to depict a dash in morse code.The LED and the buzzer are in sync with each other. DigitalWrite allows the led to blink and buzzer to beep in high and then low with 3*unit delay for a dash.

```
void dash()
{
Serial.print("-");
digitalWrite(led, HIGH);
digitalWrite(buz, HIGH);
delay(unit_delay * 3);
digitalWrite(led, LOW);
digitalWrite(buz, LOW);
delay(unit_delay);
}
```

We use a function for each letter and call the dots and dash functions as per the respective morse code value.Given below is the function for the letter A.

```
void A()
{
dot();
delay(unit_delay);
dash();
delay(unit_delay);
}
```

**Decoder**

The Morse_decod function contains the morse[] array for the morse code value of the characters.

```
void Morse_decod()
{
static String morse[] = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....",
"..", ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-",
".-.", "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--..", "!"
};
```

The MakeString function facilitates taking user input. If the button is pressed for less than 0.6 sec, it's considered as a dot, else it's considered as a dash.

```
char MakeString()
{
if (pres_len < (unit_delay*3) && pres_len > 50)
{
return '.';                    //if button press less than 0.6sec, it is a dot
}
else if (pres_len > (unit_delay*3))
{
return '-';                    //if button press more than 0.6sec, it is a dash
}
}
```

The setup function is used to set the button as input and LED as output. The Baud Rate is 9600.

```
void setup() {
Serial.begin(9600);
pinMode(but, INPUT_PULLUP);
pinMode(led, OUTPUT);
}
```

## 2.3 Proposed System

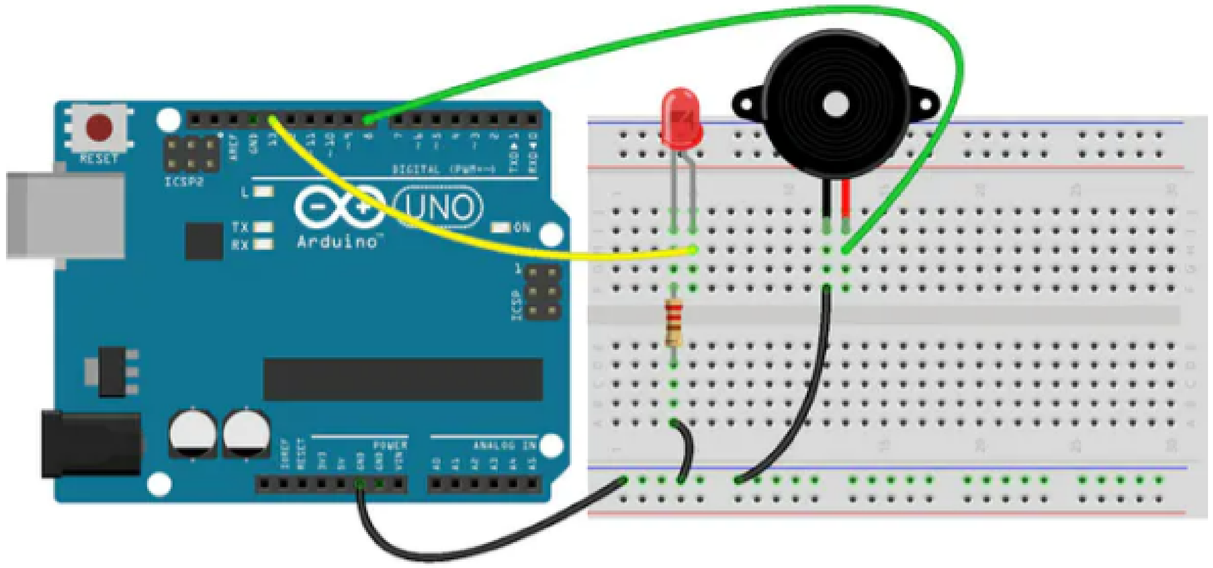


Figure 2.3(a) Encoder circuit

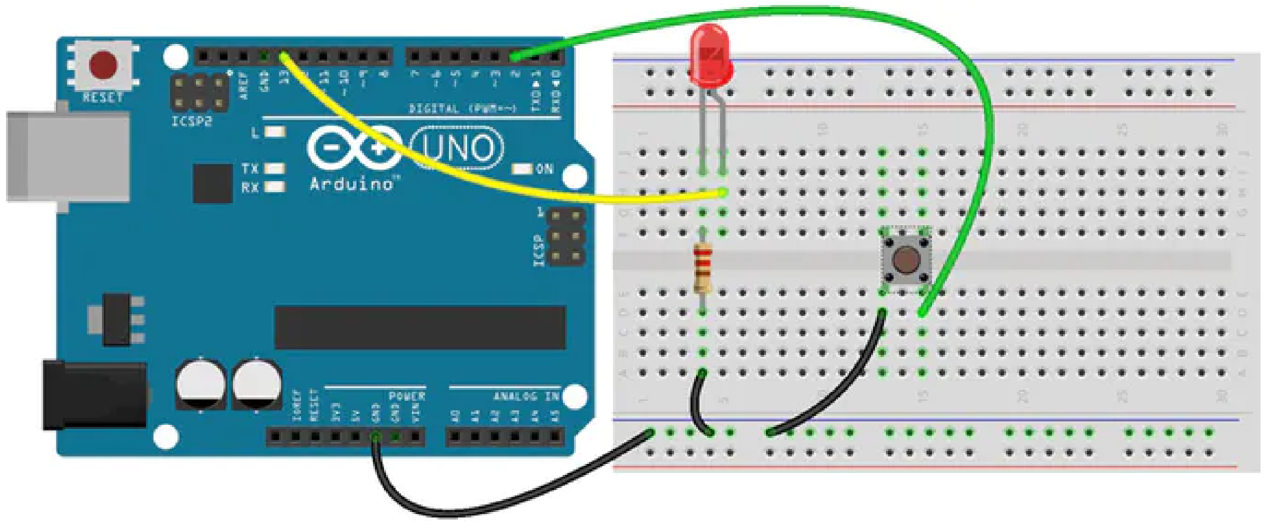


Figure 2.3(b) Decoder circuit

### 2.3.1 Economic Feasibility

.

The project used the following components:

1. Arduino UNO R3
2. Cable
3. Breadboard
4. Jumper Wires
5. LED
6. Resistor
7. Buzzer
8. Push Button
9. Laptop

The average cost of the above mentioned components come around Rs. 1000 excluding the cost of a laptop. However, the cost can be brought down if the purchases are made in bulk.

The total amount of time spent on this project was around 3 months, including studying about the topic, acquiring all components, assembling them and designing the code.

### 2.3.2 Technical feasibility:

The components, when assembled properly, make two seperate circuits, one of the encoder and the other of the decoder. The project is feasible as all the components are available and the technology exists. It is a practical project and can be done with a team of three.

The software used in the project was Arduino IDE 1.8.16. The open-source Arduino Software (IDE) made it easy to write code and upload it to the board. This software can be used with any Arduino board.

The use of LED makes the project efficient for people with hearing disability and the use of Buzzer makes the project efficient for the people with vision disability. Hence, our purpose is fulfilled.

### 2.3.3 Operational feasibility

The team members had to work online due to the pandemic and the hardware was only assembled by one member, due to which a few problems were faced.
The components were not easily available in the neighbourhood.
The morse code is outdated so it'll be rarely used, as other systems like braille exist.
Morse code was used as an international standard for maritime distress until 1999 when it was replaced by the Global Maritime Distress and Safety System.

## 2.4 Overview of software

Arduino IDE 1.8.16
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.
Active development of the Arduino software is hosted by GitHub.
In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

## 2.5 Hardware Specification

| | |
|---|---|
| Arduino UNO R3 | x 1 |

The Arduino Uno Rev 3 is a microcontroller board based on the ATmega328,
an 8-bit microcontroller with 32KB of Flash memory and 2KB of RAM with 14 digital input/output pins,6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

| | |
|---|---|
| ElectroPeak Active Buzzer | x 1 |

Active buzzer, no frequency signal, only need to offer signal to level can sound, Suitable for button prompt, prompt warning and so on.

LED x 1

A light-emitting diode is a semiconductor light source that emits light when current flows through it.

ElectroPeak Jumper wire x 1

A jump wire is an electrical wire,, with a connector or pin at each end which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally without soldering.

Breadboard x 1

A breadboard is a rectangular plastic board with a bunch of tiny holes in it. These holes let you easily insert electronic components to prototype an electronic circuit.

Resistor x 1

A resistor is a passive electrical component with the primary function to limit the flow of electric current.

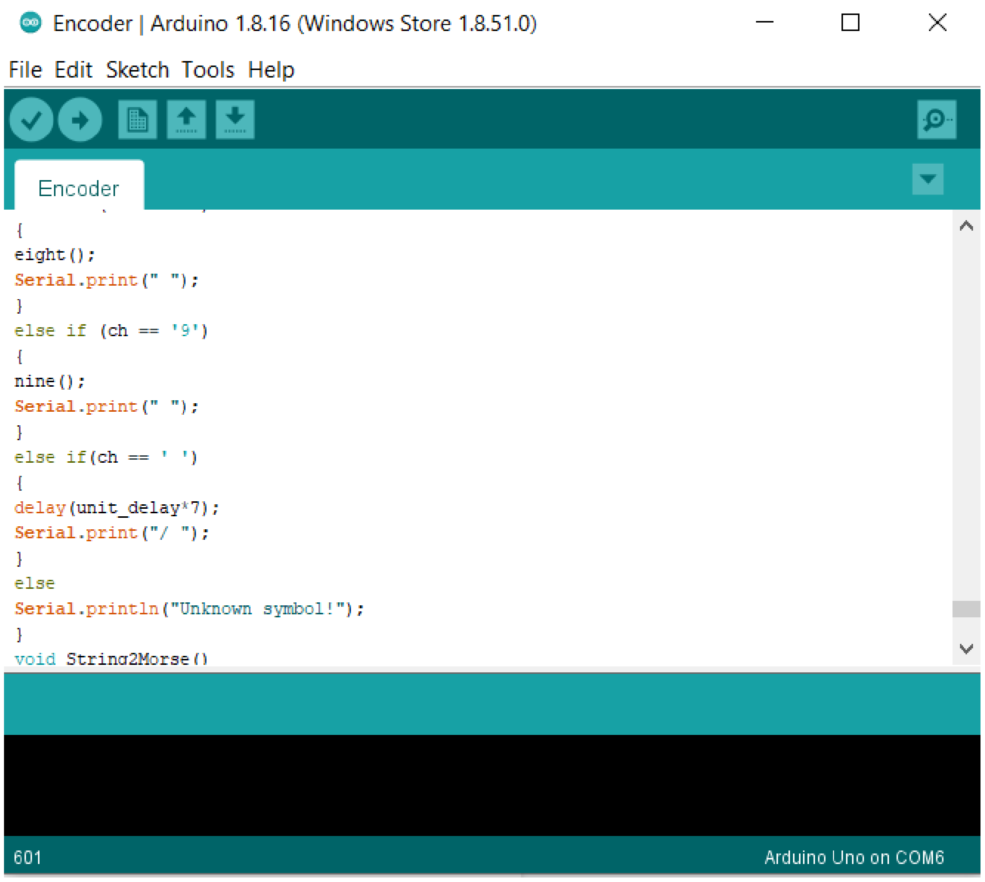
Push Button x 1

A push-button is a simple switch mechanism to control some aspect of a machine or a process.

## 2.6 Software Requirements

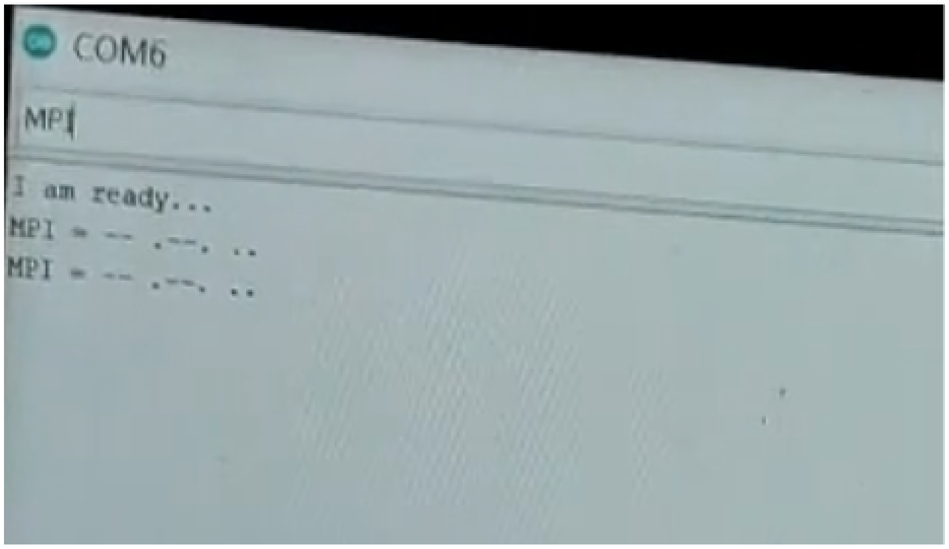
Arduino is an open source electronics platform which uses simple I/O boards and a development environment to interact with the board. Arduino boards are capable of performing multiple functionalities based on the set of instructions sent to the microcontroller. The open-source Arduino software (Arduino IDE) is used to write code (C and C++) and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board.

https://www.arduino.cc/en/software


Encoder | Arduino 1.8.16 (Windows Store 1.8.51.0)
File Edit Sketch Tools Help
Encoder

```
{
eight();
Serial.print(" ");
}
else if (ch == '9')
{
nine();
Serial.print(" ");
}
else if(ch == ' ')
{
delay(unit_delay*7);
Serial.print("/ ");
}
else
Serial.println("Unknown symbol!");
}
void String2Morse()
```

Figure 2.6(a) Software Environment



```
I am ready...
MPI - -- .--. ..
MPI - -- .--. ..
```

Figure 2.6(b) Serial Monitor Output

## 2.7 Summary

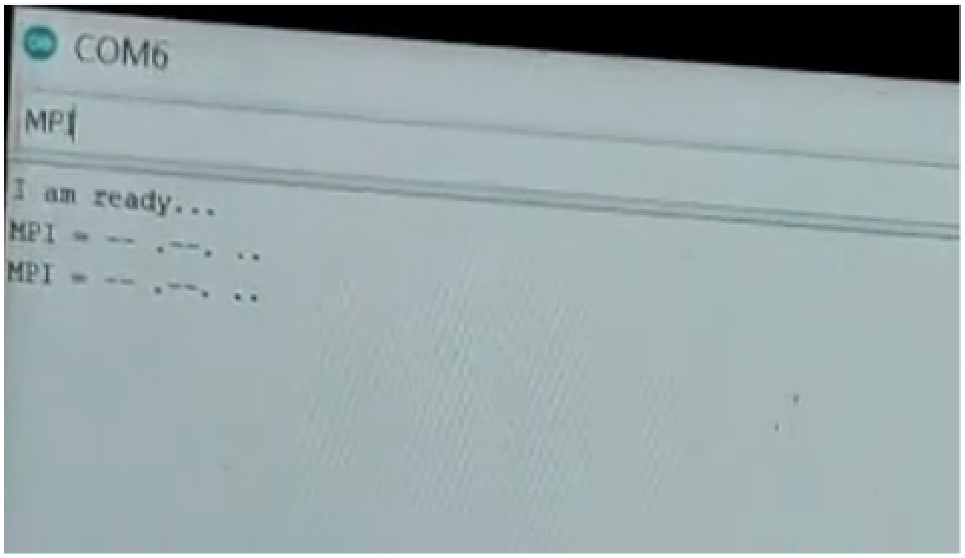
Hence, the project idea can be implemented for a basic demonstration. The idea behind this project is simply to use Arduino to flash out a message of our choice via an LED and beep out the same via a buzzer. If the input is in English, then the output is in Morse COde and vice versa. Arduino applications can be written using C or C++. The dot function and the dash function are used to depict a dot and a dash respectively in morse code. The LED and the buzzer are in sync with each other. The blink of the LED and the beep of the buzzer occur in high and low with 3*unit delay for a dash and unit delay for a dot. The code was developed with a few trials and subsequently, the circuit was designed. The economical, technical and operational feasibilities of the project were reasonable. The costs excluding the cost of the computer could be covered. Even though some components were not easily available, the technical aspects were also fulfilled. Once the hardware components were purchased, the circuit could be set up easily. The hardware and software requirements were met. Arduino IDE 1.8.16 was used for this project. The open-source Arduino Software (IDE) allowed writing the code and uploading it to the board. This is a practical project and was completed by three members in a span of three months.

# CHAPTER 3

# Result and Analysis / Testing

The setup is made ready with the breadboard connected to an LED and a buzzer. The Arduino IDE is opened to run the code. The computer is now connected to the Arduino using the USB cable and the code is uploaded using the Arduino compiler. When an English word is written on the serial monitor in the encoder circuit, it is converted to Morse Code. The buzzer generates the Morse Code sound since the pin for the buzzer is connected to the Arduino. The LED also blinks in long and short sets that denote dashes and dots respectively. It can be noticed that the length of a dash is around three times the length of a dot. The length of the dots and dashes as well as the length of the time between each character can be adjusted in the code. In the decoder circuit, the push button is pressed in long and short sets that denote dashes and dots respectively to give the input in Morse code. The output is displayed in English in the serial monitor.

## 3.1 Encoder



```
I am ready...
MPI = -- .--. ..
MPI = -- .--. ..
```

**Figure 3.1(a)**

**The encoder output on the Serial monitor**

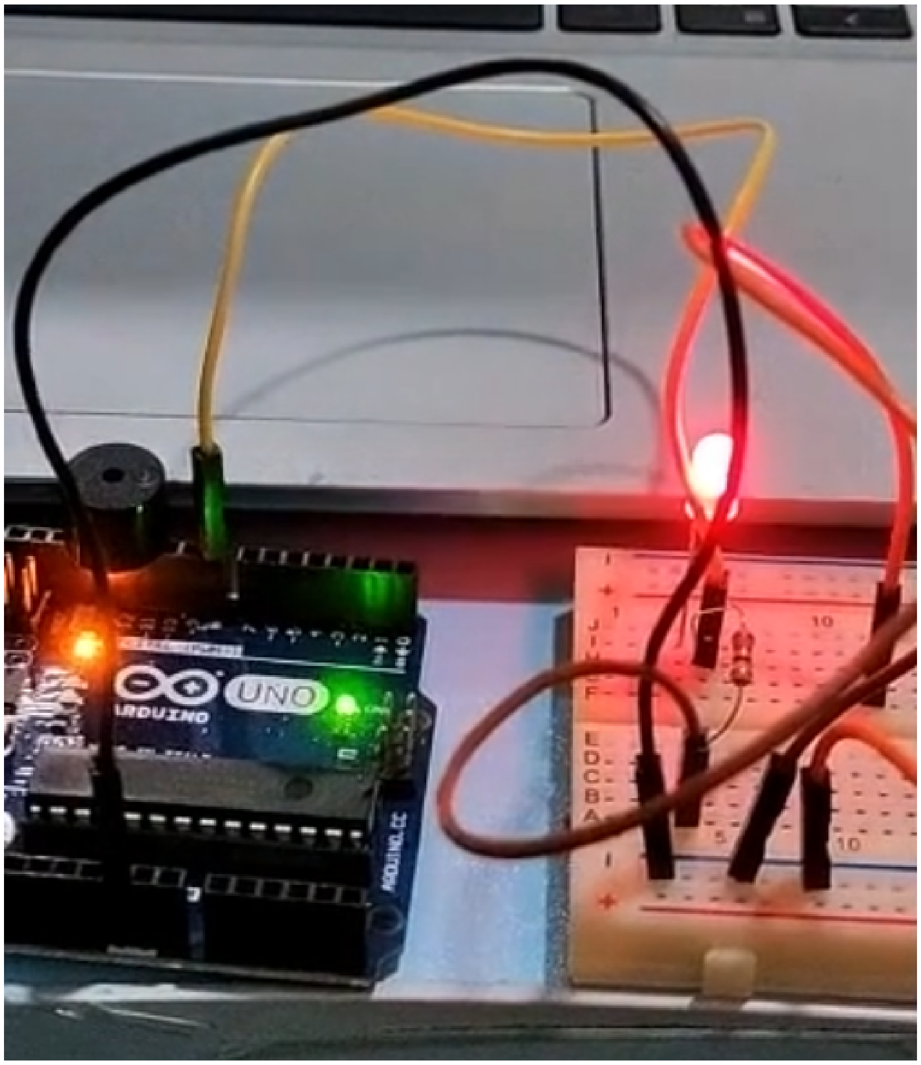


**Figure 3.1(b)**

**The encoder circuit with the buzzer and LED**

## 3.2 Decoder:

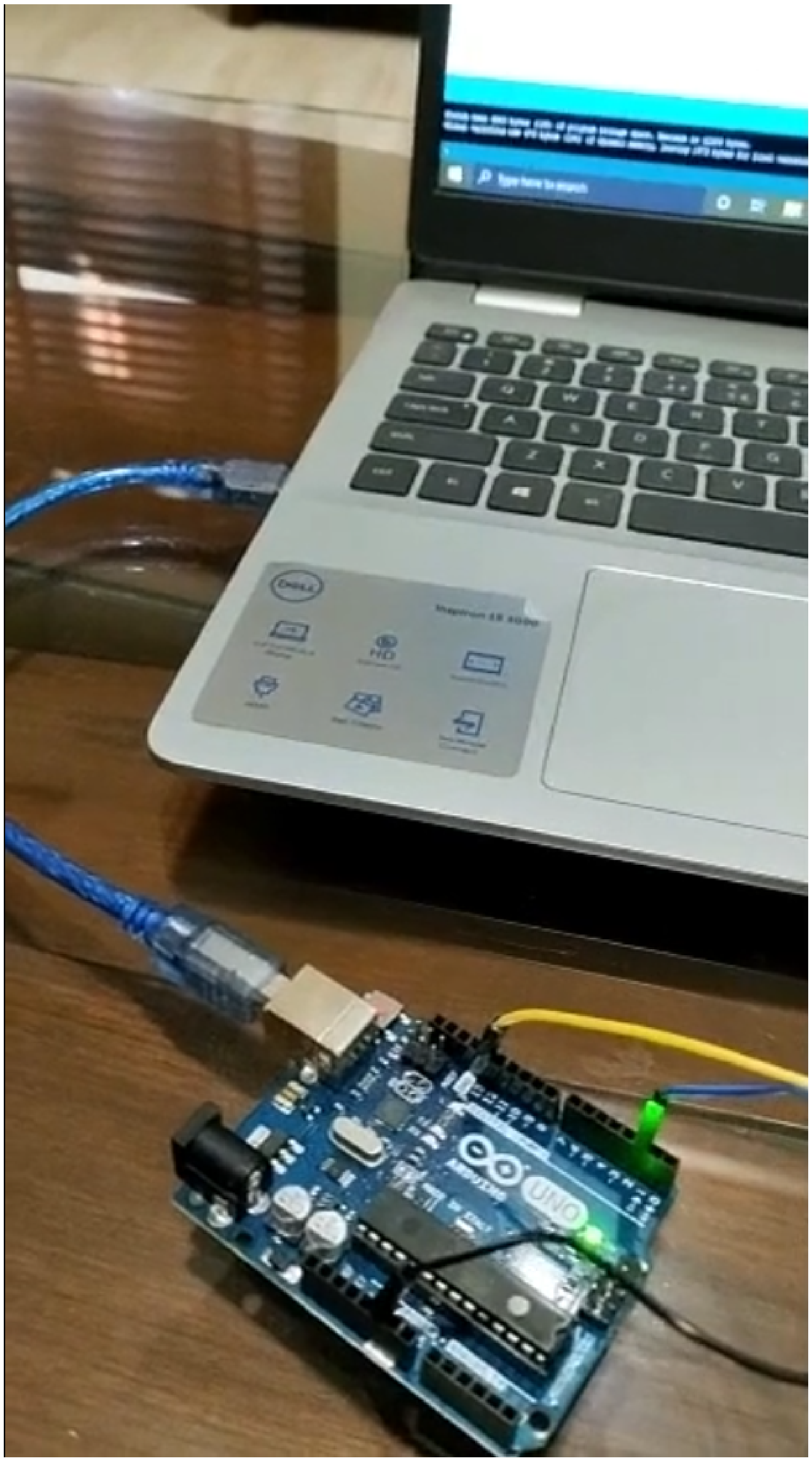


**Figure 3.2(a)**

**Arduino Connected with Laptop**

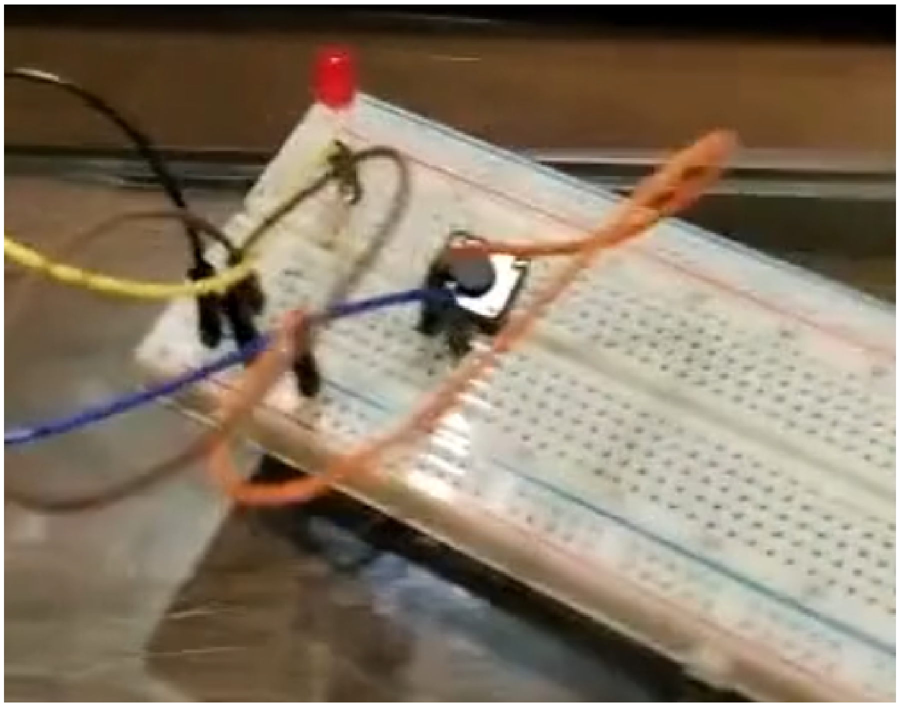


**Figure 3.2(b)**

**Breadboard with the button and LED for decoder**

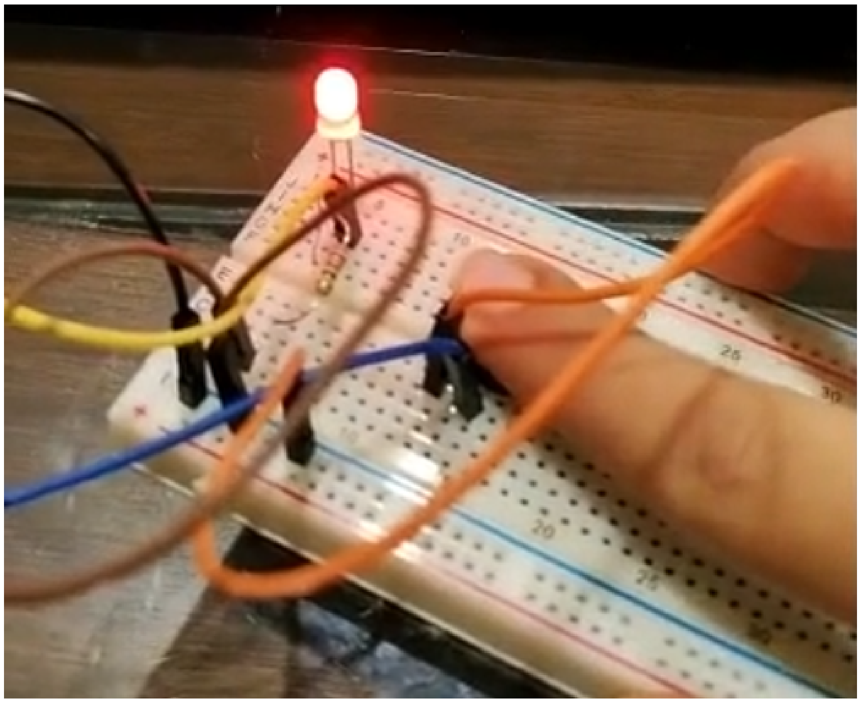
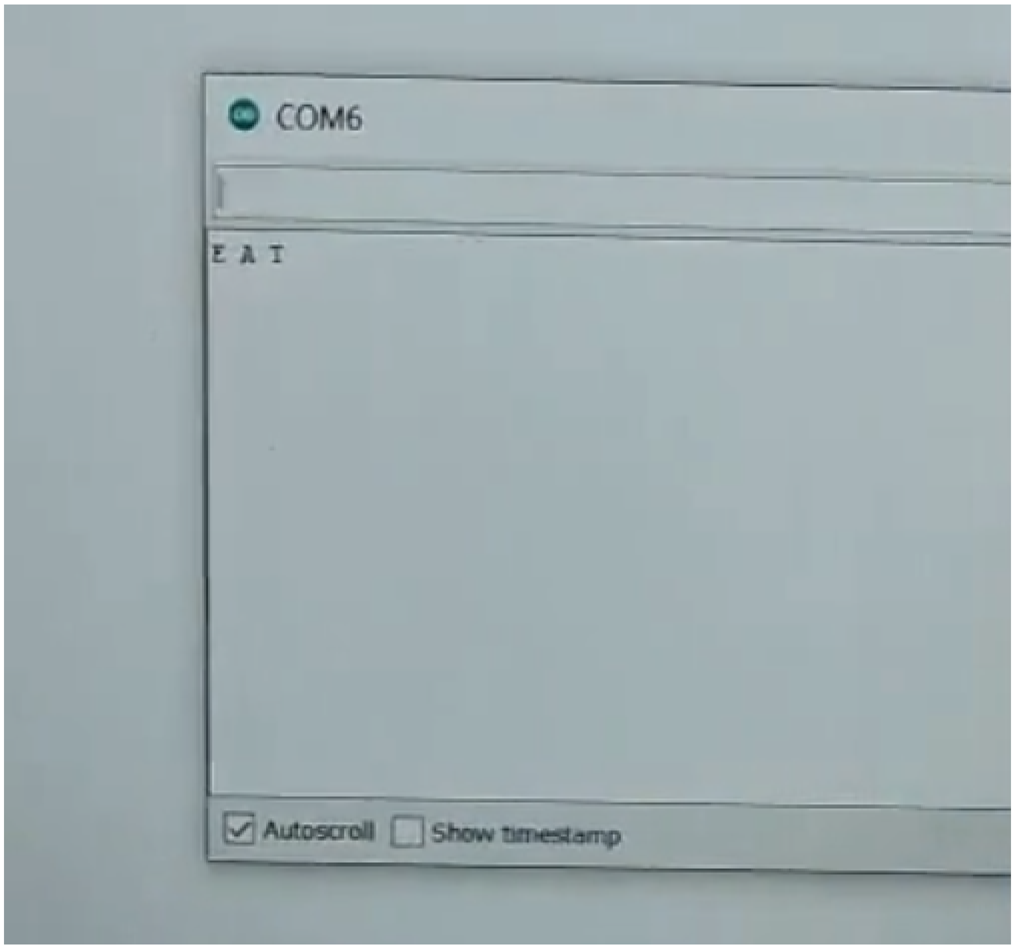


**Figure 3.3(c)**

**The glowing LED when the button is pressed**



```
E A T
```

**Figure 3.2(d)**

**Decoded Output on Serial Monitor**

## 3.3 Summary

Hence, we successfully implemented the morse code encoder and decoder using Arduino UNO R3. It can be used as an efficient medium of communication by the deaf, blind and dumb. Furthermore, we can use it to send and receive encrypted messages.

The components, when assembled properly, make two seperate circuits, one of the encoder and the other of the decoder. The project is feasible as all the components are available and the technology exists.

# CHAPTER 4

# CONCLUSION AND FUTURE ENHANCEMENT

This project shows successful implementation of the Morse Code Convertor. Morse Code can be used by the disabled people to communicate with each other. Since its signals are so simple - on or off, long or short - Morse code can also be used by flashing lights. A receiver would read the flashes and convert it back to text.

Since radio navigational aids such as VORs and NDBs still identify in Morse Code, it is also most commonly used in aviation and aeronautical fields today. People with disabilities or those harmed by a stroke, heart attack, or paralysis have used Morse Code for communication. There have been multiple instances where people could communicate in Morse Code using their eyelids - lengthy and quick blinks to indicate dots and dashes. The wires, magnets, and keys used in the initial demonstration of Samuel's Morse Code have inspired the on-screen keyboards in today's smartphones. Morse code is no longer just a telegraph language of dots and dashes, but it has also proven to be a real boon.

# APPENDIX

## ENCODER CODE

```
/*

	Morse code - Transmitter

	*/

	const int led = 13;
	const int buz = 8;

	String code = "";
	int len = 0;
	char ch;
	char new_char;
	int unit_delay = 250;


	void dot()
	{
	 Serial.print(".");
	 digitalWrite(led, HIGH);
	 digitalWrite(buz, HIGH);
	 delay(unit_delay);
	 digitalWrite(led, LOW);
	 digitalWrite(buz, LOW);
	 delay(unit_delay);
	}
```

```
void dash()
{
 Serial.print("-");
 digitalWrite(led, HIGH);
 digitalWrite(buz, HIGH);
 delay(unit_delay * 3);
 digitalWrite(led, LOW);
 digitalWrite(buz, LOW);
 delay(unit_delay);
}


void A()
{
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void B()
{
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
```

```
void C()
{
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void D()
{
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void E()
{
 dot();
 delay(unit_delay);
}
void f()
{
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void G()
{
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void H()
{
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void I()
{
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
```

```
}
void J()
{
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void K()
{
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void L()
{
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
```

```
}
void M()
{
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void N()
{
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void O()
{
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void P()
{
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
}
void Q()
{
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void R()
{
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void S()
{
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
```

```
}
void T()
{
 dash();
 delay(unit_delay);
}
void U()
{
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void V()
{
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void W()
{
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void X()
{
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void Y()
{
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void Z()
{
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void one()
{
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void two()
{
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
```

```
}
void three()
{
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void four()
{
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void five()
{
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void six()
{
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void seven()
{
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void eight()
{
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void nine()
{
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dot();
 delay(unit_delay);
}
void zero()
```

```
{
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
 dash();
 delay(unit_delay);
}
void morse()
{
 if (ch == 'A' || ch == 'a')
 {
   A();
   Serial.print(" ");
 }
 else if (ch == 'B' || ch == 'b')
 {
   B();
   Serial.print(" ");
 }
 else if (ch == 'C' || ch == 'c')
 {
   C();
   Serial.print(" ");
 }
 else if (ch == 'D' || ch == 'd')
 {
   D();
   Serial.print(" ");
 }
 else if (ch == 'E' || ch == 'e')
 {
   E();
   Serial.print(" ");
 }
 else if (ch == 'f' || ch == 'f')
 {
   f();
   Serial.print(" ");
 }
 else if (ch == 'G' || ch == 'g')
 {
   G();
   Serial.print(" ");
 }
 else if (ch == 'H' || ch == 'h')
 {
   H();
   Serial.print(" ");
 }
 else if (ch == 'I' || ch == 'i')
 {
   I();
   Serial.print(" ");
 }
 else if (ch == 'J' || ch == 'j')
 {
   J();
   Serial.print(" ");
 }
 else if (ch == 'K' || ch == 'k')
 {
   K();
   Serial.print(" ");
 }
 else if (ch == 'L' || ch == 'l')
 {
   L();
   Serial.print(" ");
 }
 else if (ch == 'M' || ch == 'm')
 {
   M();
   Serial.print(" ");
 }
 else if (ch == 'N' || ch == 'n')
 {
   N();
   Serial.print(" ");
 }
 else if (ch == 'O' || ch == 'o')
 {
   O();
   Serial.print(" ");
 }
 else if (ch == 'P' || ch == 'p')
 {
   P();
   Serial.print(" ");
```

```
}
else if (ch == 'Q' || ch == 'q')
{
  Q();
  Serial.print(" ");
}
else if (ch == 'R' || ch == 'r')
{
  R();
  Serial.print(" ");
}
else if (ch == 'S' || ch == 's')
{
  S();
  Serial.print(" ");
}
else if (ch == 'T' || ch == 't')
{
  T();
  Serial.print(" ");
}
else if (ch == 'U' || ch == 'u')
{
  U();
  Serial.print(" ");
}
else if (ch == 'V' || ch == 'v')
{
  V();
  Serial.print(" ");
}
```

```
else if (ch == 'W' || ch == 'w')
{
  W();
  Serial.print(" ");
}
else if (ch == 'X' || ch == 'x')
{
  X();
  Serial.print(" ");
}
else if (ch == 'Y' || ch == 'y')
{
  Y();
  Serial.print(" ");
}
else if (ch == 'Z' || ch == 'z')
{
  Z();
  Serial.print(" ");
}
else if (ch == '0')
{
  zero();
  Serial.print(" ");
}
else if (ch == '1')
{
  one();
  Serial.print(" ");
}
else if (ch == '2')
```

```
{
  two();
  Serial.print(" ");
}
else if (ch == '3')
{
  three();
  Serial.print(" ");
}
else if (ch == '4')
{
  four();
  Serial.print(" ");
}
else if (ch == '5')
{
  five();
  Serial.print(" ");
}
else if (ch == '6')
{
  six();
  Serial.print(" ");
}
else if (ch == '7')
{
  seven();
  Serial.print(" ");
}
else if (ch == '8')
{
   eight();
   Serial.print(" ");
  }
 else if (ch == '9')
  {
   nine();
   Serial.print(" ");
  }
 else if(ch == ' ')
  {
   delay(unit_delay*7);
   Serial.print("/ ");
  }
 else
   Serial.println("Unknown symbol!");
}


void String2Morse()
{
 len = code.length();
 for (int i = 0; i < len; i++)
  {
   ch = code.charAt(i);
   morse();
  }
}


void setup() {
 Serial.begin(9600);
 pinMode(led, OUTPUT);
 pinMode(buz, OUTPUT);
 Serial.println("I am ready...");
}


void loop() {
 while (Serial.available())
 {
   code = Serial.readString();
   Serial.print(code);
   Serial.print(" = ");
   String2Morse();
   Serial.println("");
 }
 delay(1000);
}
```

**DECODER CODE**

```
/*

           Morse code - Receiver

           */



           String code = "";
           int len = 0;
           char ch;
           char new_char;
```

```
const int but = 2;
const int led = 13;
unsigned long pres_len = 0, rel_time, pres_time = 0, old_time_len = 0, old_pres = 0,
space = 0;
int state = 0;
int unit_delay = 250;
int min_delay = 10;


char MakeString()
{
 if (pres_len < (unit_delay*3) && pres_len > 50)
 {
  return '.';                    //if button press less than 0.6sec, it is a dot
 }
 else if (pres_len > (unit_delay*3))
 {
  return '-';                    //if button press more than 0.6sec, it is a dash
 }
}


void Morse_decod()
{
 static String morse[] = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....",
 "..", ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-",
 ".-.", "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--..", "!"
};
int i = 0;

while (morse[i] != "!")
{
 if (morse[i] == code)
 {
   Serial.print(char('A' + i));
   Serial.print(" ");
   break;
 }
 i++;
}


if (morse[i] == "!")
{
 Serial.println("");
 Serial.println("This code is not exist!");
}
code = "";
}


void setup()
{
 Serial.begin(9600);
 pinMode(but, INPUT_PULLUP);
 pinMode(led, OUTPUT);
}
```

```
void loop()
{
 label:
 while (digitalRead(but) == HIGH) {}
 old_pres = rel_time;
 pres_time = millis();
 digitalWrite(led, HIGH);
 while (digitalRead(but) == LOW) {}
 rel_time = millis();
 digitalWrite(led, LOW);
 pres_len = rel_time - pres_time;
 space = pres_time - old_pres;
 if (pres_len > min_delay)
 {
   code += MakeString();
 }
 while ((millis() - rel_time) < (unit_delay * 3))
 {
   if (digitalRead(but) == LOW)
   {
     goto label;
   }
 }

 Morse_decod();
}
```

# REFERENCES
**(Standard IEEE Format Sample)**

- Deaf-Vibe: A Vibrotactile Communication Device Based on Morse Code for Deaf-Mute Individuals

Alex Rupom Hasdak;Istiaq Al Nur;Adnan Al Neon;Hasan U. Zaman
2018 9th IEEE Control and System Graduate Research Colloquium (ICSGRC)
Year: 2018 | Conference Paper | Publisher: IEEE

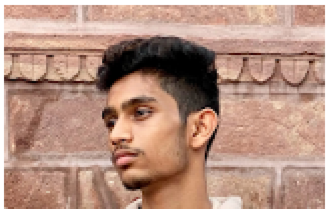
- Novel concept of bionic Morse coding for mimicry covert underwater communication

Muhammad Bilal;Songzuo Liu;Gang Qiao;Waleed Raza;Habib Hussain Zuberi
2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)
Year: 2020 | Conference Paper | Publisher: IEEE

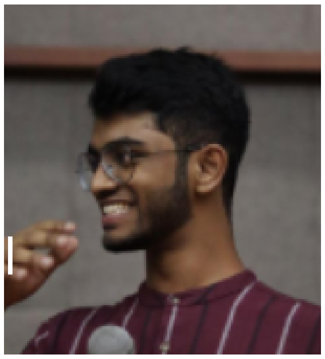
- Morse code generator using Microcontroller with alphanumeric keypad
  Paparao Nalajala;Bhavana Godavarth;M Lakshmi Raviteja;Deepthi Simhadri

2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)
Year: 2016 | Conference Paper | Publisher: IEEE

## BIODATA



Name: Bhavishya Kumar
Mobile Number: +91 9884380004
E-mail : bhavishya.kumar2020@vitstudent.ac.in



Name : Josiah James
Mobile Number: 8976183483
E-mail : josiahjgeorge@gmail.com

Name: Ayesha Numa
Mobile Number : 7397534398
E-mail : ayeshanuma13@gmail.com