**ELEC 291 Section 20C**

**LAB #2: MORSE CODE WITH ARDUINO**

**LAB SECTION: L2D**

**TEAM #: D\_5C**

|  |  |  |
| --- | --- | --- |
| ***Student name*** | ***Student number*** | ***Contribution percentage*** |
| **LINDA MUNISI** | **22535158** | **33.3%** |
| **SIRINE TRIGUI** | **46443157** | **33.3%** |
| **YIXIN ZHAO** | **49427140** | **33.3%** |

*Contribution summary*:

In the lab we worked on the first part of the code that included entering text or characters and printing them in the serial monitor. While we all contributed ideas to the code, we had Yixin and Sirine typing the code on the screen while Linda connected the circuit on the Arduino. We met a couple of times to work on the lab together since we have one circuit, making it difficult to do it alone at home. When we met we roughly divided the work a follows, Linda wrote the function of receiving message and the Fritzing diagram. Yixin and Sirine wrote the function for displaying letters and numbers in the 7-Segment display. We all contributed in coding the Morse code, green LED, 7-Segment LED displaying and the report.

**A. Introduction and motivations**

In the lab described below, we used some new elements, Piezo buzzer and 7-segment LED, to build an audio and a visible display of Morse code. This time more effort was put into coding instead of wiring. This lab introduced basic system design which is essential for further study.

In order to build the system as above, we implemented functions receiving input from the keyboard, translating letters, numbers and punctuations to Morse code. Besides, we displayed the message on 7-Segment LED and the Morse code through LED visibly as well as audibly through Piezo Buzzer.

The motivation for this lab was the desire of seeing the displayed characters and numbers on the 7-Segment LED. We have been using LED’s in previous classes but did not experience how they were made to produce the numbers when typed. Therefore the excitement to make this happen was our motivation. In addition, an interest for the course also motivates us in doing the labs.

**B. Lab Description**

As far as we considered, this lab was divided into three parts. The circuit, the functions that takes input, lighting of the green LED, translation of characters and numbers to Morse code, and the displaying through 7-Segment LED. We first started with a simple basic circuit that consisted of a green LED connected through the resistor to the ground. Then, we proceeded to write the functions.

After searching the functions in Arduino, we agreed to use Serial.read() to receive the input. First we used this to receive the value of WPM that we desired to be used in our program. Our WPM function prompts the user to give us a number as the wanted WPM. We implemented it as inputFromUser() and called it in the loop(). Then we used the same function in a different implementation to get the message that the user enters from the serial monitor. A user is prompted to input WPM, and then the WPM is set to the value that the user entered. Then the user is prompted to enter a message and prints the corresponding number in form of ASCII code on the monitor.

Then we wired our circuit. Since the 7-Segment display has an internal negative logic, we had to be careful when we were encoding. The translation was fairly easy too. We used the same skeleton code for all the letters and just coded it according to the table of the Morse code. After we got the Morse code working it was easy to implement the buzzer by adding the tone function that made the sound to the buzzer. We controlled the volume of the buzzer by changing the value that we passed in the function.

When we were coding for displaying letters, numbers and full-stop on 7-Segment LED in the first attempt, we used duplicated code that differs only depending on what the letter or the number was. After talking with our instructor, we decided to write a general function for this task. We tried to directly code it in binary but it was not taken by the program as binary but a very large number instead. Finally, we changed it to take one hexadecimal number and by masking the number, it gives the condition of each segment. One thing we need to be careful about, for masking it, we need to use “&” instead of “&&”.

In summary our code works as follows, after the code is uploaded the user is prompted to enter a value of WPM. At this point, all the components (buzzer, LED and Seven Segment LED are off). Then after the WPM is set, a user is prompted to enter the message. The serial monitor afterwards prints it in form of ASCII characters. While doing that, the following is also done simultaneously;

* The green LED lights translating the dots and dashes of the Morse code of the character entered.
* The buzzer produces a sound that is simultaneous to the lighting on and off of the green LED, it produces a sound when on and no sound when the LED is off.
* The seven segment displays the character or number that has been entered.

An additional observation is that the 7-Segment LED does not display punctuation other than full-stop despite them being displayed in the green LED and buzzer using their Morse code. Also when nothing is entered in the serial monitor the seven Segment LED is always off.

**C. Conclusions**

From this lab we learnt that, when any character or number is typed in the Keyboard, it is converted into computer-readable form using the ASCII code. Each letter, number or punctuation has its own representation in ASCII code. Using the conversion table was useful to determine if the code functionality was right.

We also realised that, after having an understanding of Morse code, it is possible to have functions to make the Morse code display characters on the seven segment, light the LED or produce a sound depending on the Morse code signal generated. The foundation of the lab was knowing how to implement Morse code and how each character, number or letter is represented using dots and dashes

Generally, we got a good understanding on the basic and advanced functionalities of Arduino using Morse code. We are now able to use Morse code not only to display words and letters but we can also make a desirable sound or tone of our choice based on the letters and characters as each produces a different tone depending on its Morse Code. The knowledge of Morse code in using Arduino proves to be of great importance in the electronics field.

**D. References**

Below are the references that we used for the report:

* Masking the bits when implementing the function for the seven segment LED: <http://stackoverflow.com/questions/10090326/how-to-extract-specific-bits-from-a-number-in-c>
* For the functions that control the tone for the buzzer;

<https://www.arduino.cc/en/reference/tone>

* Understanding how to implement the function that controls the PWM:

<https://www.arduino.cc/en/Tutorial/PWM>

* Observing the demonstration of how the Morse code signal was supposed to result to, we used an example of the SOS signal from Wikipedia;

<https://en.wikipedia.org/wiki/SOS>

* To implement the Morse code for characters, letters and Punctuations we used the chart for International Morse code from Wikipedia;

<https://en.wikipedia.org/wiki/Morse_code#/media/File:International_Morse_Code.svg> along with the chart provided in the in-class notes for referencing the Morse code for punctuation and Error or Erase.

* To test if the characters that we type are the correct ones being displayed in the serial monitor in form of ASCII codes, we used the ASCII control code chart from Wikipedia; <https://en.wikipedia.org/wiki/ASCII>
* For converting the resistors from color code to values or vice-versa we used an online resistor calculator;

<http://www.electronics2000.co.uk/calc/resistor-code-calculator.php>

* The datasheets that we referred to are;

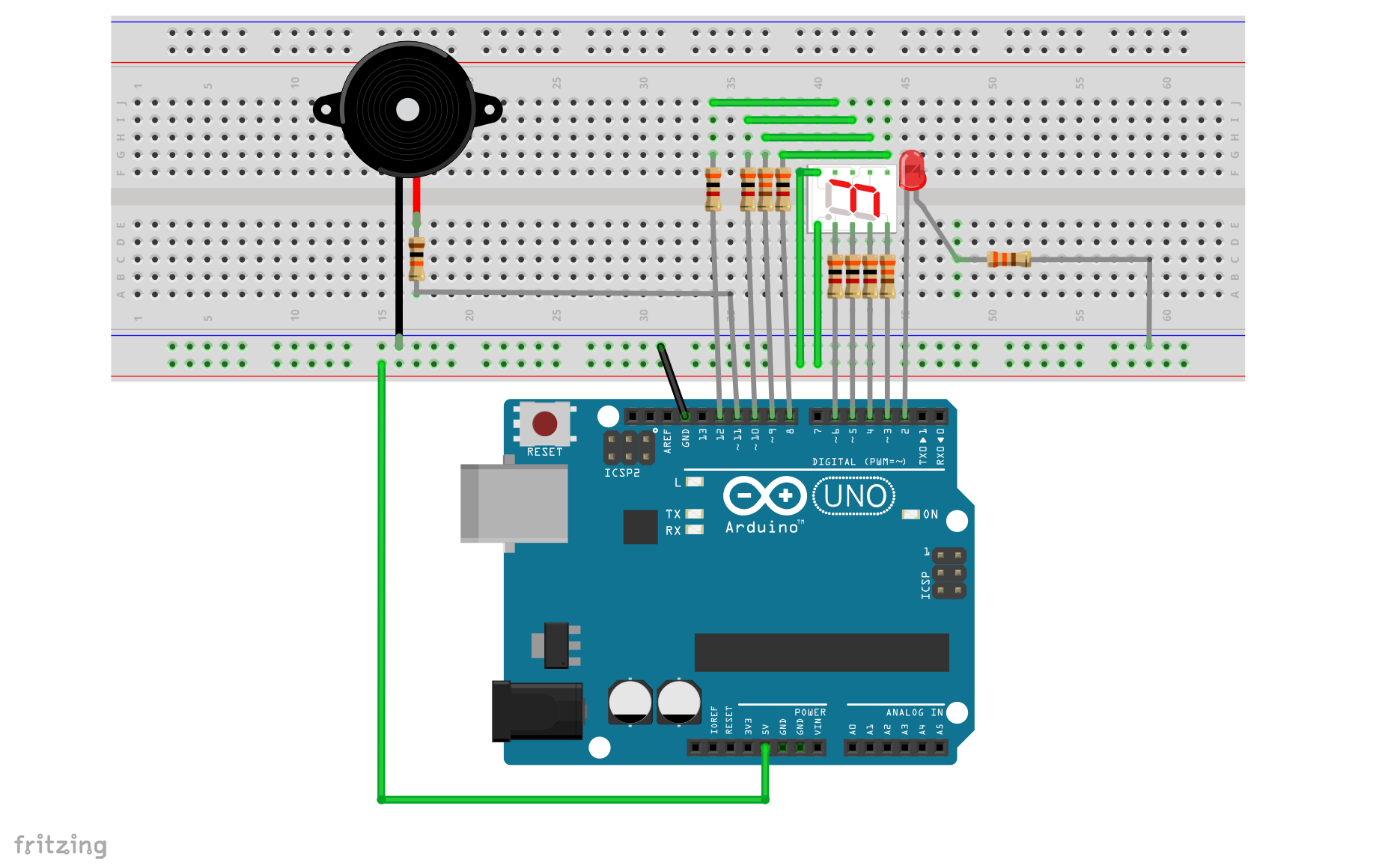
Piezo Buzzer - [pz\_datasheet.pdf](https://connect.ubc.ca/bbcswebdav/pid-3136867-dt-content-rid-14226996_1/courses/SIS.UBC.ELEC.291.20C.2015W2.59753/ELEC291_15W2/L2/291_L2_LabInfo_images/pz_datasheet.pdf) and, for the 7-Segment;

<http://optoelectronics.liteon.com/upload/download/DS30-2002-151/S_110_LSHD-F101.pdf>

**APPENDICES:**

**Appendix I**

Below is a snapshot of our Fritzing breadboard schematic:



**Appendix II**

int ledPin = 2; // the pin that the LED is attached to

//connecting the 7seg to arduino’s board

int forA = 3; // se

int forB = 4;

int forC = 5;

int forDP = 6;

int forD = 12;

int forE = 10;

int forF = 8;

int forG = 9;

int forBuzzer = 11;//buzzer connected to pin 11(PWM pin)

int incomingByte; // a variable to read incoming serial data into

int WPM = 5; // initializing WPM

int dotLength = 1200/WPM; // getting dotLength for morse code

int dashLength = dotLength \* 3; // getting dashLength for morse code

int flag = 1;

int flag2 = 1;

//general function to display the letters and numerals

void lightforall(int number){

int array[8];

int x = 0x80;

//each array element contains 0 or 1 depending on whether the specified seg is on or off

// array elements encoded to represent the segments as following: a,b,c,d e,f,g,DP

for(int i=0;i<=7;i++){

array[i] = number & x;

x = x >>1;

}

//if array[0]= 0 , seg a will be off

if(array[0]== 0x00){

digitalWrite(forA, HIGH);

}else{

digitalWrite(forA, LOW); //otherwise seg a will be on

}

if(array[1]== 0x00){

digitalWrite(forB, HIGH);

} else{

digitalWrite(forB, LOW);

}

if(array[2]== 0x00){

digitalWrite(forC, HIGH);

} else{

digitalWrite(forC, LOW);

}

if(array[3]== 0x00){

digitalWrite(forD, HIGH);

} else{

digitalWrite(forD, LOW);

}

if(array[4]== 0x00){

digitalWrite(forE, HIGH);

} else{

digitalWrite(forE, LOW);

}

if(array[5]== 0x00){

digitalWrite(forF, HIGH);

} else{

digitalWrite(forF, LOW); }

if(array[6]== 0x00){

digitalWrite(forG, HIGH);

} else{

digitalWrite(forG, LOW);

}

if(array[7]== 0x00){

digitalWrite(forDP, HIGH);

} else{

digitalWrite(forDP, LOW); delay(2000);}

}

//setting the LED on

void on() {

digitalWrite(ledPin, HIGH);

}

//setting the LED off

void off() {

digitalWrite(ledPin, LOW);

}

//lightening up the LED for dotLength and setting the buzzer for dotLength

void dot() {

on();

analogWrite(forBuzzer,LOW);

tone(forBuzzer,700);

//noTone(forBuzzer);

delay(dotLength);

analogWrite(forBuzzer, HIGH); }

//lightening up the LED for dashLength and setting the buzzer for dashLength

void dash() {

on();

analogWrite(forBuzzer, LOW);

tone(forBuzzer,700);

// noTone(forBuzzer);

delay(dotLength \* 3);

analogWrite(forBuzzer, HIGH);

}

//turning off the LED to represent the space

void space() {

off();

delay(dotLength);

}

//resting for dashLength between each letter

void btnLetters() {

off();

delay(dotLength \*3);

}

//restting for 7\*dotLength between each word

void btnWords() {

off();

delay(dotLength \* 7);

}

void SevenSegLEDOff(){

lightforall(0x00);

}

//displaying letters

void lightA(){

lightforall(0xEE);

off();

}

void lightB(){

lightforall(0x3E);

off(); }

void lightC(){

lightforall(0x9C);

off(); }

void lightD(){

lightforall(0x7A);

off(); }

void lightE(){

lightforall(0x9E);

off();}

void lightF(){

lightforall(0x8E);

off();}

void lightG(){

lightforall(0xF6);

off(); }

void lightH(){

lightforall(0x2E);

off();}

void lightI(){

lightforall(0x0C);

off();}

void lightJ(){

lightforall(0x78);

off();}

void lightK(){

lightforall(0x6E);

off(); }

void lightL(){

lightforall(0x1C);

off(); }

void lightM(){

lightforall(0xA8);

off();}

void lightN(){

lightforall(0x2A);

off(); }

void lightO(){

lightforall(0xFC);

off(); }

void lightP(){

lightforall(0xCE);

off();}

void lightQ(){

lightforall(0xE6);

off();}

void lightR(){

lightforall(0x0A);

off(); }

void lightS(){

lightforall(0xB6);

off();}

void lightT(){

lightforall(0x1E);

off();}

void lightU(){

lightforall(0x7C);

off();}

void lightV(){

lightforall(0x38);

off();}

void lightW(){

lightforall(0x54);

off();}

void lightX(){

lightforall(0x6E);

off();}

void lightY(){

lightforall(0x76);

off();}

void lightZ(){

lightforall(0xDA);

off();}

//displaying the dot

void lightDot(){

lightforall(0x01);

off();

}

// Methods for numbers:

void lightNum0(){

lightforall(0xFC);

off();}

void lightNum1(){

lightforall(0x60);

off();}

void lightNum2(){

lightforall(0xDA); off();}

void lightNum3(){

lightforall(0xF2); off();}

void lightNum4(){

lightforall(0x66); off(); }

void lightNum5(){

lightforall(0xB6); off(); }

void lightNum6(){

lightforall(0x3E); off(); }

void lightNum7(){

lightforall(0xE0); off(); }

void lightNum8(){

lightforall(0xFE); off();}

void lightNum9(){

lightforall(0xF6); off();}

//prompting the user to enter WPM value

void inputFromUser(){

Serial.print("Enter the WPM:");

delay(500);

//enabling the user to enter the value from keyboard

WPM = Serial.parseInt();

delay(200);

Serial.println(WPM);

}

void setup() {

// initialize serial communication:

Serial.begin(9600);

// initialize the LED pin as an output:

pinMode(ledPin, OUTPUT);

pinMode(forA, OUTPUT);

pinMode(forB, OUTPUT);

pinMode(forC, OUTPUT);

pinMode(forD, OUTPUT);

pinMode(forE, OUTPUT);

pinMode(forF, OUTPUT);

pinMode(forG, OUTPUT);

pinMode(forDP, OUTPUT);

}

void loop() {

SevenSegLEDOff();

//WPM to be entered

if (flag2 == 1 ){

inputFromUser();

delay(2000);

flag2 =0;}

if(flag== 1){

Serial.println("Enter the message:");

flag=0;

}

// checking for incoming serial data:

if (Serial.available() > 0) {

// read the oldest byte in the serial buffer:

incomingByte = Serial.read();

Serial.print("You have entered: ");

Serial.println(incomingByte, DEC); //this prints the ASCII code for whatever has been printed in the monitor

//encoding letters in Morse code:

if (incomingByte == 'A' || incomingByte == 'a') {

lightA(); dot(); space(); dash(); btnLetters();

}

if (incomingByte == 'B' || incomingByte == 'b') {

lightB(); dash(); space(); dot(); space(); dot(); space(); dot(); btnLetters();

}

if (incomingByte == 'C' || incomingByte == 'c') {

lightC(); dash(); space(); dot(); space(); dash(); space(); dot(); btnLetters();

}

if (incomingByte == 'D' || incomingByte == 'd') {

lightD(); dash(); space(); dot(); space(); dot(); btnLetters();

}

if (incomingByte == 'E' || incomingByte == 'e'){

lightE(); dot(); btnLetters();

}

if (incomingByte == 'F' || incomingByte == 'f') {

lightF(); dot(); space(); dot(); space(); dash(); space(); dot(); btnLetters(); }

}

if (incomingByte == 'G' || incomingByte == 'g') {

lightG(); dash(); space(); dash(); space(); dot(); btnLetters();

}

if (incomingByte == 'H' || incomingByte == 'h') {

lightH(); dot(); space(); dot(); space(); dot(); space(); dot(); btnLetters();

}

if (incomingByte == 'I' || incomingByte == 'i') {

lightI(); dot(); space(); dot(); btnLetters();

}

if (incomingByte == 'J' || incomingByte == 'j') {

lightJ(); dot(); space(); dash(); space(); dash(); space(); dash(); btnLetters();

}

if (incomingByte == 'K' || incomingByte == 'k'){

lightK(); dash(); space(); dot(); space(); dash(); btnLetters();

}

if (incomingByte == 'L' || incomingByte == 'l'){

lightL(); dot(); space(); dash(); space(); dot(); space(); dot(); btnLetters();

}

if (incomingByte == 'M' || incomingByte == 'm') {

lightM(); dash(); space(); dash(); btnLetters();

}

if (incomingByte == 'N' || incomingByte == 'n') {

lightN(); dash(); space(); dot(); btnLetters();

}

if (incomingByte == 'O' || incomingByte == 'o') {

lightO(); dash(); space(); dash(); space(); dash(); btnLetters();

}

if (incomingByte == 'P' || incomingByte == 'p') {

lightP(); dot(); space(); dash(); space(); dash(); space(); dot(); btnLetters();

}

if (incomingByte == 'Q' || incomingByte == 'q') {

lightQ(); dash(); space(); dash(); space();dot(); space(); dash(); btnLetters();

}

if (incomingByte == 'R' || incomingByte == 'r') {

lightR(); dot(); space(); dash(); space(); dot(); btnLetters();

}

if (incomingByte == 'S' || incomingByte == 's'){

lightS();dot(); space(); dot(); space(); dot(); btnLetters();

}

if (incomingByte == 'T' || incomingByte == 't') {

lightT();dash(); btnLetters();

}

if (incomingByte == 'U' || incomingByte == 'u') {

lightU(); dot(); space(); dot(); space(); dash(); btnLetters();

}

if (incomingByte == 'V' || incomingByte == 'v') {

lightV();dot(); space(); dot(); space(); dot(); space(); dash(); btnLetters();

}

if (incomingByte == 'W' || incomingByte == 'w') {

lightW();dot(); space(); dash(); space(); dash(); btnLetters();

}

if (incomingByte == 'X' || incomingByte == 'x') {

lightX(); dash(); space(); dot(); space(); dot(); space(); dash(); btnLetters();

}

if (incomingByte == 'Y' || incomingByte == 'y') {

lightY();dash(); space(); dot(); space(); dash(); space(); dash(); btnLetters();

}

if (incomingByte == 'Z' || incomingByte == 'z') {

lightZ();dash(); space(); dash(); space(); dot(); space(); dot(); btnLetters();

}

//encoding numbers in Morse code

if (incomingByte == '0') {

lightNum0();dash(); space();dash(); space();dash(); space(); dash(); space(); dash(); btnLetters(); }

if (incomingByte == '1'){

lightNum1();dot(); space(); dash(); space();dash(); space(); dash(); space(); dash(); btnLetters(); }

if (incomingByte == '2'){

lightNum2();dot(); space(); dot(); space(); dash(); space(); dash(); space(); dash(); btnLetters(); }

if (incomingByte == '3'){

lightNum3();dot(); space(); dot(); space(); dot(); space(); dash(); space(); dash(); btnLetters(); }

if (incomingByte == '4'){

lightNum4();dot(); space(); dot(); space(); dot(); space(); dot(); space(); dash(); btnLetters(); }

if (incomingByte == '5'){

lightNum5();dot(); space(); dot(); space(); dot(); space(); dot(); space(); dot(); btnLetters(); }

if (incomingByte == '6'){

lightNum6();dash(); space(); dot(); space(); dot(); space(); dot(); space(); dot(); btnLetters(); }

if (incomingByte == '7'){

lightNum7();dash(); space(); dash(); space(); dot(); space(); dot(); space(); dot(); btnLetters(); }

if (incomingByte == '8'){

lightNum8();dash(); space(); dash(); space(); dash(); space(); dot(); space(); dot(); btnLetters(); }

if (incomingByte == '9'){

lightNum9();dash(); space(); dash(); space(); dash(); space(); dash(); space(); dot(); btnLetters(); }

//encoding the four punctuation in Morse code

if ( incomingByte == '.') {

lightDot(); dot(); space(); dash(); space(); dot(); space(); dash(); space(); dot(); space(); dash(); }

if (incomingByte == ','){

dash(); space(); dash(); space(); dot(); space(); dot(); space(); dash(); space(); dash();}

if (incomingByte == ':'){

dash(); space(); dash(); space(); dash(); space(); dot(); space(); dot(); space(); dot(); }

if (incomingByte == '-'){

dash(); space(); dot(); space(); dot(); space(); dot(); space(); dot(); space(); dash();}

}