INTELLECTUAL PROPERTY INDIA

सत्यमेव जयते

Extracts from the Register of Copyrights

प्रतिलिप्यधिकार कार्यालय, भारत सरकार | Copyright Office, Government Of India

दिनांक/Dated:21/11/2024

1.	पंजीकरण संख्या/Registration Number	:	**SW-19668/2024**
2.	आवेदक का नाम, पता तथा राष्ट्रीयता Name, address and nationality of the applicant	:	BIRLA INDUSTRIAL AND TECHNOLOGICAL MUSEUM , 19A, GURUSADAY ROAD KOLKATA-700019 INDIAN
3.	कृति के प्रतिलिप्यधिकार में आवेदक के हित की प्रकृति Nature of the applicant's interest in the copyright of the work	:	OWNER
4.	कृति का वर्ग और वर्णन Class and description of the work	:	COMPUTER SOFTWARE WORK SOFTWARE
5.	कृति का शीर्षक Title of the work	:	MORSE CODE DETECTOR AND GENERATOR
6.	कृति की भाषा Language of the work	:	C, DOT NET
7.	रचयिता का नाम, पता और राष्ट्रीयता तथा यदि रचयिता की मृत्यु हो गई है, तो मृत्यु की तिथि Name, address and nationality of the author and if the author is deceased, date of his decease	:	SUBHABRATA CHAUDHURI , BIRLA INDUSTRIAL & TECHNOLOGICAL MUSEUM, 19A, GURUSADAY ROAD-700019 INDIAN
8.	कृति प्रकाशित है या अप्रकाशित Whether the work is published or unpublished	:	PUBLISHED
9.	प्रथम प्रकाशन का वर्ष और देश तथा प्रकाशक का नाम, पता और राष्ट्रीयता Year and country of first publication and name, address and nationality of the publisher	:	2023 INDIA BIRLA INDUSTRIAL AND TECHNOLOGICAL MUSEUM , 19A, GURUSADAY ROAD KOLKATA-700019 INDIAN
10.	बाद के प्रकाशनों के वर्ष और देश, यदि कोई हों, और प्रकाशकों के नाम, पते और राष्ट्रीयताएँ Years and countries of subsequent publications, if any, and names, addresses and nationalities of the publishers	:	N.A.
11.	कृति में प्रतिलिप्यधिकार सहित विभिन्न अधिकारों के स्वामियों के नाम, पते और राष्ट्रीयताएं और समनुदेशन और अनुज्ञप्तियों के विवरण के साथ प्रत्येक के अधिकार का विस्तार, यदि कोई हो। Names, addresses and nationalities of the owners of various rights comprising the copyright in the work and the extent of rights held by each, together with particulars of assignments and licences, if any	:	BIRLA INDUSTRIAL AND TECHNOLOGICAL MUSEUM , 19A, GURUSADAY ROAD KOLKATA-700019 INDIAN
12.	अन्य व्यक्तियों के नाम, पते और राष्ट्रीयताएं यदि कोई हों, जो प्रतिलिप्यधिकार वाले अधिकारों को समनुदेशित करने या अनुज्ञप्ति देने के लिए अधिकृत हों Names, addresses and nationalities of other persons, if any, authorised to assign or licence of rights comprising the copyright	:	N.A.
13.	यदि कृति एक 'कलात्मक कृति' है, तो कृति पर अधिकार रखने वाले व्यक्ति का नाम, पता और राष्ट्रीयता सहित मूल कृति का स्थान। (एक वास्तुशिल्प कृति के मामले में कृति पूरी होने का वर्ष भी दिखाया जाना चाहिए) If the work is an 'Artistic work', the location of the original work, including name, address and nationality of the person in possession of the work. (In the case of an architectural work, the year of completion of the work should also be shown).	:	N.A.
14.	यदि कृति एक 'कलात्मक कृति' है जो किसी भी माल या सेवाओं के संबंध में उपयोग की जाती है या उपयोग किए जाने में सक्षम है, तो आवेदन में प्रतिलिप्यधिकार अधिनियम, 1957 की धारा 45 की उप–धारा (i) के प्रावधान के अनुसार व्यापार चिह्न रजिस्ट्रार से प्रमाणन शामिल होना चाहिए। If the work is an 'Artistic work' which is used or capable of being used in relation to any goods or services, the application should include a certification from the Registrar of Trade Marks in terms of the provision to Sub-Section (i) of Section 45 of the Copyright Act, 1957.	:	N.A.
15.	यदि कृति एक 'कलात्मक कृति' है, तो क्या यह डिजाइन अधिनियम 2000 के अंतर्गत पंजीकृत है? यदि हां, तो विवरण दें। If the work is an 'Artistic work', whether it is registered under the Designs Act 2000, if yes give details.	:	N.A.
16.	यदि कृति एक 'कलात्मक कृति' है, जो डिजाइन अधिनियम 2000 के तहत एक डिजाइन के रूप में पंजीकृत होने में सक्षम है, तो क्या यह औद्योगिक प्रक्रिया के माध्यम से किसी वस्तु पर प्रयुक्त की गई है और यदि हाँ, तो इसे कितनी बार पुनरुत्पादित किया गया है? If the work is an 'Artistic work', capable of being registered as a design under the Designs Act 2000.whether it has been applied to an article though an industrial process and ,if yes ,the number of times it is reproduced.	:	N.A.
17.	टिप्पणी यदि कोई हो/Remarks, if any	:	NONE

डायरी संख्या/Diary Number: 9532/2024-CO/SW

आवेदन की तिथि/Date of Application: 28/03/2024

प्राप्ति की तिथि/Date of Receipt: 28/03/2024

उमाशंकर पंडित

Registrar of Copyrights

COPY OF FIRST 10 PAGES

Project Name: Morse Code Detector and Generator

Computer Side Codes:

1. Form1.cs file:

```
//-----------------------------------Start of Form1.cs-----------------------------------//
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.IO.Ports;
using System.Linq;
using System.Reflection;
using System.Security.Policy;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ArtefactSW
{
  public partial class Form1 : Form
  {
    private int counter;
    private int timervalue;
    public static System.IO.Ports.SerialPort port;
    delegate void SetTextCallback(string text);

    // This BackgroundWorker is used to demonstrate the
    // preferred way of performing asynchronous operations.
    private BackgroundWorker bardWorker;

    private Thread readThread = null;
    private bool flagtimer = false;
    public Form1()
    {
      InitializeComponent();
      bardWorker = new BackgroundWorker();
    }

    private void SetText(string val)
    {
      // InvokeRequired required compares the thread ID of the
      // calling thread to the thread ID of the creating thread.
      // If these threads are different, it returns true.
      if (this.txtarduinovalue.InvokeRequired)
      {
        SetTextCallback d = new SetTextCallback(SetText);
        this.Invoke(d, new object[] { val });
      }
      else
      {
        //this.txtarduinovalue.Text += "Text: ";
        if (val == "Resetting\r" || val == "Ready\r")
        {
        }
        else
```

```
        {
            //this.txtarduinovalue.Text += val;
            //this.txtarduinovalue.Text += Environment.NewLine;
            if (val.Contains('>')) // Text message
            {
                txtoriginal.Text = txtoriginal.Text + val.Substring(1, 1);
                //timer1.Start();
                flagtimer = true;
                counterfunction();
            }
            else  // Morse Code
            {
              //string str = val.TrimEnd('\r'); // remove special character
              //if(str.Length > 0) // to check
              //{
              //    txtsymbol.Text = txtsymbol.Text + " " + str;
              //}
              txtsymbol.Text = txtsymbol.Text + " " + val;
            }
        }
        txtband.Focus();
    }
}
private void connection()
{
    System.ComponentModel.IContainer components =
        new System.ComponentModel.Container();
    port = new System.IO.Ports.SerialPort(components);
    port.PortName = ddlcomport.SelectedItem.ToString();
    port.BaudRate = 9600,//Int32.Parse(baudRate.SelectedItem.ToString());
    port.DtrEnable = true;
    port.ReadTimeout = 5000;
    port.WriteTimeout = 500;
    port.Open();

    readThread = new Thread(new ThreadStart(this.Read));
    readThread.Start();
    this.hardWorker.RunWorkerAsync();
}
public void Read()
{
    while (port.IsOpen)
    {
        try
        {
            if (port.BytesToRead > 0)
            {
                string message = port.ReadLine();
                this.SetText(message);
            }
            else
            {

            }
        }
        catch (TimeoutException) { }
    }

private void Form1_FormClosed(object sender, FormClosedEventArgs e)
```

```
{
  try
  {
    if (!(readThread == null))
      readThread.Abort();
  }
  catch (NullReferenceException)
  {
  }

  try
  {
    port.Close();
  }
  catch (NullReferenceException)
  {
  }
}

private void timer1_Tick(object sender, EventArgs e)
{
  //if (flagtimer == true)
  //{
  //  flagtimer = false;
  //  timer1.Stop();
  //  txtoriginal.Text = "";
  //  txtsymbol.Text = "";
  //}

    counter--;
    if (counter == 0)
    {
      timer1.Stop();
      txtoriginal.Text = "";
      txtsymbol.Text = "";
      txtsend.Text = "";
      txtsend.Focus();
      //port.BaseStream.Flush();
    }
  lblcounter.Text = counter.ToString();
}

private void Form1_Load_1(object sender, EventArgs e)
{
  try
  {
    foreach (string s in SerialPort.GetPortNames())
    {
      ddlcomport.Items.Add(s);
    }
    if (ddlcomport.Items.Count > 0)
      ddlcomport.SelectedIndex = ddlcomport.Items.Count - 1;
    else
      ddlcomport.SelectedIndex = 0;
    txtcomport.Text = ddlcomport.SelectedItem.ToString();
    connection();
    paneloutput.Visible = false;
    using (StreamReader sr = new StreamReader("idletime.txt"))
    {
      timevalue = Convert.ToInt32(sr.ReadToEnd());
```

```
            }

            axWindowsMediaPlayer1.URL = Application.StartupPath + "\\Video\\Video1.mp4";
            axWindowsMediaPlayer1.settings.autoStart = true;
            axWindowsMediaPlayer1.settings.setMode("loop", true);
            axWindowsMediaPlayer1.uiMode = "None";
            axWindowsMediaPlayer1.settings.volume = 100;


          }
          catch (Exception ex)
          {
            MessageBox.Show(ex.Message);
          }
        }
        private void txtsend_KeyDown(object sender, KeyEventArgs e)
        {
          if (e.KeyCode == Keys.Enter)
          {
            if (port.IsOpen)
            {
              port.Write(txtsend.Text);
            }
          }
        }
        private void counterfunction()
        {
          counter = timervalue;

          //timer1 = new System.Windows.Forms.Timer();
          //timer1.Tick += new EventHandler(timer1_Tick);
          //timer1.Interval = 1000; // 1 second
          timer1.Start();
          lblcounter.Text = counter.ToString();
        }
      }

    }

    //----------------------------------------End of Form1.cs----------------------------------------//
```

2. Form1.Designer.cs file:

```
//----------------------------------Start of Form1.Designer.cs----------------------------------//
namespace ArtefactSW
{
  partial class Form1
  {
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
    protected override void Dispose(bool disposing)
    {
      if (disposing && (components != null))
      {
        components.Dispose();
      }
      base.Dispose(disposing);
    }

    #region Windows Form Designer generated code

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      this.components = new System.ComponentModel.Container();
      System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
      this.panneloutput = new System.Windows.Forms.Panel();
      this.ddlcomport = new System.Windows.Forms.ComboBox();
      this.label4 = new System.Windows.Forms.Label();
      this.label2 = new System.Windows.Forms.Label();
      this.txtarduinovalue = new System.Windows.Forms.TextBox();
      this.txtcomport = new System.Windows.Forms.TextBox();
      this.txtoriginal = new System.Windows.Forms.TextBox();
      this.txtsymbol = new System.Windows.Forms.TextBox();
      this.label1 = new System.Windows.Forms.Label();
      this.label3 = new System.Windows.Forms.Label();
      this.txtsend = new System.Windows.Forms.TextBox();
      this.label5 = new System.Windows.Forms.Label();
      this.label6 = new System.Windows.Forms.Label();
      this.timer1 = new System.Windows.Forms.Timer(this.components);
      this.lblcounter = new System.Windows.Forms.Label();
      this.label7 = new System.Windows.Forms.Label();
      this.axWindowsMediaPlayer1 = new AxWMPLib.AxWindowsMediaPlayer();
      this.panneloutput.SuspendLayout();
      ((System.ComponentModel.ISupportInitialize)(this.axWindowsMediaPlayer1)).BeginInit();
      this.SuspendLayout();
      //
      // panneloutput
      //
      this.panneloutput.Controls.Add(this.ddlcomport);
      this.panneloutput.Controls.Add(this.label4);
```

```
this.paneloutput.Controls.Add(this.label2);
this.paneloutput.Controls.Add(this.txtarduinovalue);
this.paneloutput.Controls.Add(this.txtcomport);
this.paneloutput.Location = new System.Drawing.Point(1009, 339);
this.paneloutput.Name = "paneloutput";
this.paneloutput.Size = new System.Drawing.Size(243, 330);
this.paneloutput.TabIndex = 14;
//
// ddlcomport
//
this.ddlcomport.FormattingEnabled = true;
this.ddlcomport.Location = new System.Drawing.Point(117, 293);
this.ddlcomport.Name = "ddlcomport";
this.ddlcomport.Size = new System.Drawing.Size(121, 21);
this.ddlcomport.TabIndex = 18;
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(22, 29);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(73, 13);
this.label4.TabIndex = 17;
this.label4.Text = "Arduino Value";
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(34, 267);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(50, 13);
this.label2.TabIndex = 15;
this.label2.Text = "Com Port";
//
// txtarduinovalue
//
this.txtarduinovalue.Location = new System.Drawing.Point(119, 26);
this.txtarduinovalue.Multiline = true;
this.txtarduinovalue.Name = "txtarduinovalue";
this.txtarduinovalue.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
this.txtarduinovalue.Size = new System.Drawing.Size(98, 225);
this.txtarduinovalue.TabIndex = 13;
//
// txtcomport
//
this.txtcomport.Location = new System.Drawing.Point(117, 267);
this.txtcomport.Name = "txtcomport";
this.txtcomport.Size = new System.Drawing.Size(100, 20);
this.txtcomport.TabIndex = 11;
//
// txtoriginal
//
this.txtoriginal.Font = new System.Drawing.Font("Microsoft Sans Serif", 15F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.txtoriginal.Location = new System.Drawing.Point(31, 175);
this.txtoriginal.Multiline = true;
this.txtoriginal.Name = "txtoriginal";
this.txtoriginal.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
this.txtoriginal.Size = new System.Drawing.Size(617, 213);
this.txtoriginal.TabIndex = 17;
```

```
            //
            // txtsymbol
            //
            this.txtsymbol.Font = new System.Drawing.Font("Microsoft Sans Serif", 15F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.txtsymbol.Location = new System.Drawing.Point(31, 469);
            this.txtsymbol.Multiline = true;
            this.txtsymbol.Name = "txtsymbol";
            this.txtsymbol.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.txtsymbol.Size = new System.Drawing.Size(617, 213);
            this.txtsymbol.TabIndex = 18;
            //
            // label1
            //
            this.label1.BackColor = System.Drawing.Color.Transparent;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.Location = new System.Drawing.Point(269, 132);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(160, 31);
            this.label1.TabIndex = 19;
            this.label1.Text = "Message";
            //
            // label3
            //
            this.label3.BackColor = System.Drawing.Color.Transparent;
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label3.Location = new System.Drawing.Point(141, 408);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(429, 44);
            this.label3.TabIndex = 20;
            this.label3.Text = "Incoming Signal / Morse Code";
            //
            // txtsend
            //
            this.txtsend.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.txtsend.Location = new System.Drawing.Point(234, 86);
            this.txtsend.Name = "txtsend";
            this.txtsend.Size = new System.Drawing.Size(414, 38);
            this.txtsend.TabIndex = 22;
            this.txtsend.KeyDown += new System.Windows.Forms.KeyEventHandler(this.txtsend_KeyDown);
            //
            // label5
            //
            this.label5.BackColor = System.Drawing.Color.Transparent;
            this.label5.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label5.Location = new System.Drawing.Point(11, 84);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(217, 47);
            this.label5.TabIndex = 19;
            this.label5.Text = "Input Message";
            //
            // label6
            //
            this.label6.BackColor = System.Drawing.Color.Transparent;
            this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
```

```
            this.label6.Location = new System.Drawing.Point(691, 81);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(401, 45);
            this.label6.TabIndex = 19;
            this.label6.Text = "Press enter to send message";
            //
            // timer1
            //
            this.timer1.Enabled = true;
            this.timer1.Interval = 1000;
            this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
            //
            // lblcounter
            //
            this.lblcounter.BackColor = System.Drawing.Color.Transparent;
            this.lblcounter.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.lblcounter.Location = new System.Drawing.Point(1108, 132);
            this.lblcounter.Name = "lblcounter";
            this.lblcounter.Size = new System.Drawing.Size(160, 31);
            this.lblcounter.TabIndex = 19;
            this.lblcounter.Text = "Counter";
            this.lblcounter.Visible = false;
            //
            // label7
            //
            this.label7.BackColor = System.Drawing.Color.Transparent;
            this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 40F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label7.Location = new System.Drawing.Point(12, 4);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(1235, 60);
            this.label7.TabIndex = 19;
            this.label7.Text = "Morse Code Generator and Detector";
            this.label7.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
            //
            // axWindowsMediaPlayer1
            //
            this.axWindowsMediaPlayer1.Enabled = true;
            this.axWindowsMediaPlayer1.Location = new System.Drawing.Point(681, 175);
            this.axWindowsMediaPlayer1.Name = "axWindowsMediaPlayer1";
            this.axWindowsMediaPlayer1.OcxState =
((System.Windows.Forms.AxHost.State)(resources.GetObject("axWindowsMediaPlayer1.OcxState")));
            this.axWindowsMediaPlayer1.Size = new System.Drawing.Size(570, 508);
            this.axWindowsMediaPlayer1.TabIndex = 21;
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackgroundImage = ((System.Drawing.Image)(resources.GetObject("$this.BackgroundImage")));
            this.BackgroundImageLayout = System.Windows.Forms.ImageLayout.None;
            this.ClientSize = new System.Drawing.Size(1280, 720);
            this.Controls.Add(this.txtsend);
            this.Controls.Add(this.axWindowsMediaPlayer1);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.lblcounter);
```

```
            this.Controls.Add(this.label1);
            this.Controls.Add(this.txtsymbol);
            this.Controls.Add(this.txtoriginal);
            this.Controls.Add(this.paneloutput);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load_1);
            this.paneloutput.ResumeLayout(false);
            this.paneloutput.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.axWindowsMediaPlayer1)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion
        private System.Windows.Forms.Panel paneloutput;
        private System.Windows.Forms.ComboBox ddlcomport;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox txtcomport;
        private System.Windows.Forms.TextBox txtoriginal;
        private System.Windows.Forms.TextBox txtsymbol;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label3;
        private AxWMPLib.AxWindowsMediaPlayer axWindowsMediaPlayer1;
        private System.Windows.Forms.TextBox txtsend;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.TextBox txtarduinovalue;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.Label lblcounter;
        private System.Windows.Forms.Label label7;
    }
}

//----------------------------------End of Form1.Designer.cs----------------------------------//
```

Microcontroller Side Codes:

3. MorseCode_Artefact.ino File:

```
//----------Start of MorseCode_Artefact.ino----------//
 //morse is based on units of time
 //1 unit length = a dit
 //3 units length = a dah
 //1 unit of length between dits and dahs
 //3 units of length between characters of a word
 //7 units of length between words in a sentence

#include <MorseCodeAbhijit1.h>

MorseCode morse = MorseCode();//create the Morse object

bool usingSerial = false;//flag for the serial input section

void setup() {
  Serial.begin(9600);//start the serial monitor
  morse.led1 = 11;//pin number of LED1
  morse.led2 = 12;//pin number of LED2
  morse.morseKey = 7;//pin number for morse code key (or momentary switch)
  morse.buzzer = 9;//pin number buzzer

  pinMode(morse.led1, OUTPUT);
  pinMode(morse.led2, OUTPUT);
  pinMode(morse.buzzer, OUTPUT);
  pinMode(morse.morseKey, INPUT_PULLUP);

  morse.unit = (110);//the length of a unit - speed, defines the length of dits dahs and spaces etc
  morse.debounceDelay = 80;//the debounce time between key press
  morse.showMorseCharacters = true;//display dit or dah after entry
  morse.showSentence = false;//display the current sentence so far
  morse.showCharacters = true;//display the current character between letters
  morse.showPhrase = false;//display certain built in phrases
  morse.speakerOut = true;//turn speaker on/off
  morse.ledOut = true;//turn leds on/off
  morse.showCharacterGap = false;//helps with the start of the next letter

  morse.resetOutput();//initialise the Morse object
}

void loop() {
  usingSerial = false;
  //get the data from the Serial window to display the character and the morse digits in the serial window
  if (Serial.available() > 0)//loop through the incoming text
  {
    usingSerial = true;
    usingSerial = morse.getMorse(Serial.read());//we have a character
    Serial.println();
  }
  if (!usingSerial)//use the key
  {
    morse.checkKeyPressType();//respond to the keying in of morse code dits and dahs
```


----------End of MorseCode_Artefact.ino----------//

COPY OF LAST 10 PAGES

4. MorseCodeAbhijit1.cpp File

```
//-----------------------------------Start of MorseCodeAbhijit1.cpp-----------------------------------//
#include "MorseCodeAbhijit1.h"
MorseCode::MorseCode()
{
        int led1;
        int led2;
        int morseKey;
        int buzzer;
        int unit;
        long debounceDelay;

        bool showMorseCharacters;
        bool showSentence;
        bool showCharacters;
        bool showPhrase;
        bool speakerOut;
        bool ledOut;
        bool showCharacterGap;

const int DIT = 1;//unit length of a dit
const int DAH = 3;//unit length of a dah
const int CODEGAP = 1; //unit length of a gap between morse dits and dahs
const int LETTERGAP = 3; //unit length of a gap between letters
const int WORDGAP = 7; //unit length of a gap between words

int buttonState;//the current reading from the input pin
int lastButtonState = LOW;// the previous reading from the input pin
long lastDebounceTime = 0;// the last time the output pin was toggled
long startCode = 0; //start time for a possible key press
long endCode = 0; //end time for a possible key press
long startTime = 0; //start time for a non-key press
long endTime = 0; //end time for a non-key press
bool gotKeyPress = false;
int previousButtonState = 0;
bool endOfWord = false;
bool foundPhrase = false;

//variables to be available on the serial monitor
String characterStore = "";//hold the current text of the input morse code
String wordStore = ""; //hold the words as they are created
String sentenceStore = ""; //holds the current sentence as it is created
String currentCharacter;//holds the current character

//work out if a key was pressed and for how long
//a one unit would be a dit and a 3 unit would be a dah
void MorseCode::checkKeyPressType()
{
        long inactivity = 0; //test for gaps in code
        int milliUnit = unit * 1.1;//modify the unit to allow for code timing
        if (debounceDelay == 0)//quick and dirty debounce
        {
                debounceDelay = 80;
        }
        buttonState = digitalRead(morseKey);//see what, if anything, is happening
        if (buttonState == LOW ) //key down
        {
                if (speakerOut)//beep if required
```

```
                {
                        analogWrite(buzzer, 180); //180 default value
                }
                if (ledOut)//glow if required (this is why we can only have one LED for the key press)
                {
                        digitalWrite(led1, HIGH);
                }
                gotKeyPress = false;
                delay(debounceDelay);//allow for key debounce
        }
        else//key up
        {
                endCode = millis();//if key now up, catch the end time
        }
        if (buttonState == HIGH && startCode != endCode) //key up after being down
        {
                analogWrite(buzzer, 0);//turn off
                digitalWrite(led1, LOW);//turn off
                inactivity = endCode - startCode;//get the time of the length of key press

                if (inactivity > WORDGAP * milliUnit * 2  && !gotKeyPress) //long press so must be a reset
                {
                        displayReset();
                        resetOutput();
                        gotKeyPress = true;
                }

                if (inactivity > milliUnit * 1.5  && !gotKeyPress) //must be a dah
                {
                        characterStore += "-";//add the dah to the store
                        if (showMorseCharacters)
                        {
                                Serial.print("-"); // to print dash on screen as input
                        }
                        resetTime();
                        gotKeyPress = true;
                }
                if (inactivity >= milliUnit * 0.2 && inactivity <= milliUnit * 1.5 && !gotKeyPress) //must be a
dit
                {
                        characterStore += ".";//add the dit to the store
                        if (showMorseCharacters)
                        {
                                Serial.print("."); // to print dot on screen as input
                        }
                        resetTime();
                        gotKeyPress = true;
                }
                resetCode();//reset the timing for the key press

        //*****************************************************************************
*****
                //got the dit or dah, now work out the gap timings for letters and words
                if (characterStore.length() > 8 )//too long (the longest is 7 + one for luck)
                {
                        characterStore = "";//reset garbage input
                }
                if (!gotKeyPress) //no point in checking this otherwise
                {
                        endTime = millis(); //capture the time of inactivity
```

```
            inactivity = endTime - startTime; //difference between start and end is how long
nothing has happened
            if (inactivity >= WORDGAP * milliUnit * 1 && !endOfWord) //end of word
            {
                    foundPhrase = checkPhrases();//see if text matches a phrase
                    if (!foundPhrase)
                    {
                            sentenceStore += wordStore + " "; //add word to sentence
                            gotKeyPress = false;
                    }
                    wordStore = ""; //end of word
                    characterStore = ""; //end of character
                    endOfWord = true;
                    doPrint(1, inactivity);//display current sentence text
                    Serial.println("> "); //Added for space
            }
            if (inactivity >= LETTERGAP * milliUnit * 1.0 && characterStore != "") //end of
character
            {
                    currentCharacter = findCharacter();//get the current character
                    if (currentCharacter != "")
                    {
                            wordStore += currentCharacter; //add the current character to the
wordstore
                            resetTime(); //reset
                            doPrint(2, inactivity);//display current sentence character
                            characterStore = ""; //reset
                            endOfWord = false;
                    }
            }
        }
    }
}

//reset variables
void MorseCode::resetOutput()
{
        resetTime();
        resetCode();
        characterStore = "";
        wordStore = "";
        sentenceStore = "";
        gotKeyPress = false;
        doPrint(0, 0);
}

//do some output
void MorseCode::doPrint(int type, int inactivity)
{
        switch (type)
        {
        case 0:
                {
                        Serial.println("Ready");
                        break;
                }
        case 1:
                if (newSentence)
                {
```

```
                    Serial.println();
                    Serial.print(" > Sentence: ");
                    Serial.println(sentenceStore);
                }
                break;
            }
        case 2:
            {
                if (showCharacters)
                {
                    Serial.println();
                    //Serial.print(" > Char: ");
                    Serial.print(">");
                    Serial.println(currentCharacter);
                }
                if(showCharacterGap)
                {
                    Serial.println(" > Character gap ");
                }
                break;
            }
        }
        //Serial.println();
    }

    //convert a boolean input to a On/Off text output
    String MorseCode::convertBool(bool value)
    {
        return value ? "On" : "Off";
    }

    //reset message
    void MorseCode::displayReset()
    {
        Serial.println();
        Serial.println();
        Serial.println("Resetting");

    }

    //reset the time for the gaps
    void MorseCode::resetTime()
    {
        startTime = millis();
        endTime = startTime;
    }

    //reset the time for the code lengths
    void MorseCode::resetCode()
    {
        startCode = millis();
        endCode = startCode;
    }
```


st of characters that we have codes for
onst char *letters =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.,?'!/()&:;=+-_\"$@ ";

odes that we have characters for, in the same order as the letters above

```
//0 = dah, 1 = dit
static const char *codes[] = {
        ".-",   //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
        "-----", //0
        ".----", //1
        "..---", //2
        "...--", //3
        "....-", //4
        ".....", //5
        "-....", //6
        "--...", //7
        "---..", //8
        "----.", //9
        ".-.-.-", //.
        "--..--", //,
        "..--..", //?
        ".----.", //'
        "-.-.--", //!
        "-..-.", ///
        "-.--.", //(
        "-.--.-", //)
        ".-...", //&
        "---...", //:
        "-.-.-.", //;
        "-...-", //=
        ".-.-.", //+
        "-....-", //-
        "..--.-", //_
        ".-..-.", //"
        "...-..-", //$
        ".--.-.", //@
        ""      //space
```


t of 34 character phrases in use

```
static const char *phrases[] = {
        "I acknowledge",                                    //QSL
        "Do you acknowledge?",          //QSL?
        "Wait",                 //QRX
        "Should I wait?",               //QRX?
        "I am ready to copy",           //QRV
        "Are you ready to copy?",       //QRV?
        "The frequency is in use",      //QRL
        "Is the frequency in use?",     //QRL?
        "My location is",               //QTH
        "What is your location?",       //QTH?
        "New line",             //AA
        "New page",             //AR
        "Wait",                 //AS
        "Break",                //BK
        "New paragraph",                //BT
        "Going off the air",                                //CL
        "Attention",            //CT
        "Change to Wabun code",         //DO
        "Invite station to transmit",                       //KN
        "End of transmission",          //SK
        "Understood",           //SN
        "Understood",           //VE
        "Distress",                             //SOS
        "Over",                 //K
        "Roger",                //R
        "See you later",                //CUL
        "Be seeing you",                //BCNU
        "You're",               //UR
        "Signal report",                //RST
        "Best regards",                 //73
        "Love and kisses",              //88
        "Error",                //HH
        "Error",                //????
        "Change to Wabun code"          //NJ
};

//the list of 34 phrase codes for above
static const char *phraseCodes[] = {
        "QSL",
        "QSL?",
        "QRX",
        "QRX?",
        "QRV",
        "QRV?",
        "QRL",
        "QRL?",
        "QTH",
        "QTH?",
        "AA",
        "AR",
        "AS",
        "BK",
        "BT",
        "CL",
        "CT",
        "DO",
        "KN",
        "SK",
        "SN",
```

```
        "VE",
        "SOS",
        "K",
        "R",
        "CUL",
        "BCNU",
        "UR",
        "RST",
        "73",
        "88",
        "HH",
        "????",
        "NJ"
};

//check if the current word matches a phrase
bool MorseCode::checkPhrases()
{
        bool phraseFound = false;
        if (showPhrase)//is it turned on
        {
                for (unsigned int code = 0; code < 34; ++code )
                {
                        if (String(phraseCodes[code]) == wordStore ) //found phrase code
                        {
                                sentenceStore += String(phrases[code]) + " ";//add the phrase to the
sentence
                                phraseFound = true;
                                break;//no need to carry on so return to calling function
                        }
                }
        }
        return phraseFound;
}

//turn the LED on/off for Serial incoming code //Serial to Morse Code Display
void MorseCode::sendToLed(int code)
{
        Serial.println();//new character
        Serial.print(">");
        Serial.println(letters[code]); // Original Serial.print(letters[code]);
        //Serial.print("  ");
        for (unsigned int i = 0; i < strlen(codes[code]); ++i )
        {
                if ( i > 0 )//put a gap between dits and dahs
                {
                        morseGap(CODEGAP);
                }
                if ( codes[code][i] == '.' )//display a dit
                {
                        morseBeep(DIT);
                        Serial.print(".");
                }
                if ( codes[code][i] == '-' )//display a dah
                {
                        morseBeep(DAH);
                        Serial.print("-");
                }
        morseGap(CHARGAP);//end of character
```

```
            //Serial.println(); //letter gap to Display
        }

        //loop through the codes to find a match
        bool MorseCode::getMorse(char character)
        {
            bool characterFound = false;
            if(character != char())
            {
                character=toUpperCase(character);
                for (unsigned int code = 0; code < strlen(letters), ++code )
                {
                    if (letters[code] == character ) //found character
                    {
                        sendToLed(code);//got the code for the character
                        characterFound = true;
                        break;//no need to carry on so return to calling function
                    }
                }
            }
            return characterFound;
        }

        //based on the dits and dahs held in characterStore, find the corresponding letter
        String MorseCode::findCharacter()
        {
            for (unsigned int i = 0; i < 55; ++i )
            {
                String code = String(codes[i]);
                String letter = String(letters[i]);
                if (code == characterStore) //current character found
                {
                    return letter;//no need to look any more
                }
            }
            return "";
        }

        //turn the LEDs off
        void MorseCode::ledOff()
        {
            digitalWrite(led1, LOW);
            digitalWrite(led2, LOW);
        }

        //define the gaps between dits, dahs, letters and words
        //for the LED
        void MorseCode::morseGap(int gapLength)
        {
            ledOff();
            delay(unit * gapLength);
        }
        //show the incoming morse dit or dah via the LED and/or the speaker
        void MorseCode::morseBeep(int unitLength)

        int currentLed=0;
        if(unitLength==DIT)//turn on the dit LED
        {
            currentLed=led1;
        }
```

```
        else//turn on the dah LED
        {
                currentLed=led2;
        }
        if (ledOut)
        {
                digitalWrite(currentLed, HIGH);
        }
        if (speakerOut)
        {
                analogWrite(buzzer, 180); //180 default value
        }
        delay(unitLength * unit);
        analogWrite(buzzer, 0);
        ledOff();
}
//----------End of MorseCodeAbhijit1.cpp----------//
```

5. Keywords.txt File


```
#######################################
# Syntax Coloring Map Morse code
#######################################

#######################################
# Class (KEYWORD1)
#######################################

Morse KEYWORD1

#######################################
# Datatypes and Variables (KEYWORD2)
#######################################

led1 KEYWORD2
led2 KEYWORD2
morseKey KEYWORD2
buzzer KEYWORD2
unit KEYWORD2
debounceDelay KEYWORD2

#######################################
# Methods and Functions (KEYWORD2)
#######################################

showMorseCharacters KEYWORD2
showSentence KEYWORD2
showCharacters KEYWORD2
showPhrase KEYWORD2
speakerOut KEYWORD2
ledOut KEYWORD2
checkKeyPressType KEYWORD2
resetOutput KEYWORD2
```

6. MorseCodeAbhijit1.h File

```
//----------Start of MorseCodeAbhijit1.h----------//

#ifndef MorseCodeAbhijit1_h
#define MorseCodeAbhijit1_h

#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#include "pins_arduino.h"
#include "WConstants.h"
#endif

class MorseCode{
public:
        MorseCode();

        int led1;//Arduino pin that the LED is connected to
        int led2;//Arduino pin that the LED is connected to
        int morseKey;//morse code key (or momentary switch)
        int buzzer;//Arduino pin that the buzzer is connected to
        int unit;//the length of a unit, defines the length of dits dahs and spaces etc
        long debounceDelay;//the debounce time
        bool showMorseCharacters;//display dit or dah after entry
        bool showSentence;//display the current sentence so far
        bool showCharacters;//display the current character between letters
        bool showPhrase;//display certain built in phrases
        bool speakerOut;//turn speaker on/off
        bool ledOut;//turn leds on/off
        bool showCharacterGap;//helps with the start of the next letter

        void checkKeyPressType();
        void resetOutput();
        bool getMorse(char character);

private:

        void doPrint(int type, int inactivity);
        String convertBool(bool value);
        void displayReset();
        void resetTime();
        void resetCode();
        bool checkPhrases();
        void sendToLed(int code);

        String findCharacter();
        void ledOn(int unitLength);
        void ledOff();
        void morse(int unitLength);
        void morseGap(int gapLength);
        void morseBeep(int unitLength);
};

//----------End of MorseCodeAbhijit1.h----------//
```