Colman O'Keeffe
114712191

Assignment 02

Module CS3514

Lecturer: Professor J. Morrison

3rd Computer Science BSc. 2016/2017

Assignment 02 Tic-Tac-Toe

Laboratory Report

1. Introduction:

    a. Project Name: Tic Tac Toe for the Arduino.

    b. Members:

        (1) Colman O'Keeffe.

        (2) Herakliusz Lipiec.

        (3) John O'Dowd.

    c. Date for Project validation: For demonstration to Lecturer and Laboratory Assistants on Tuesday 28 November 2016 at 10am during the normal lecture time.

    d. Necessary Laboratory equipment provided:

        (1) One electronics kit consisting of minimum:

            (a) One (1) x Arduino duemilanovo.

            (b) One (1) x Solderless Plug-in BreadBoard, 830 tie-points, 4 power rails, 6.5 x 2.2 x 0.3in (165 x 55 x 9mm) or similar the one listed is a commercial variant as I could not find the model of the one supplied.

            (c) Nine (9) x yellow LEDs.

            (d) Nine (9) x red LEDs.

            (e) Six (6) x Two Hundred and Twenty Ohm (220 Ω) resistors.

(f) One (1) x remote controllor (type unknown possibly generic) with battery.

(g) One (1) x IR reciever rated for 5 V.

(h) One (1) x Buzzer rated for 5 V. Alternatively use a speaker 5 V.

(i) Assorted coloured wires with breadboard pins attached. Max number needed for maximal wiring is Forty Two (42) pieces.

(j) The use of a Multimeter was required to test the resistors were correct. (Alternatively use a Resistance meter or use a Galvinometer and a known voltage).

e. Time allotted: Two x lab periods of 1 hr, in order to tie together code tested in previous labs + personal time by each of the team members.

2. Requirement:

a. As published by the lecturer https://www.dropbox.com/s/iqekeojshcd3q08/2016-10-27-CS3514-tictactoe-project.pdf?dl=0 – My apologies, I should have included this as Annex A, it was unavailable when I went to reload it, from home. Diagrams of the circuit described appear at Annex E.

b. As interpreted by the team in order to assign tasks:

(1) Build circuit to control individually Nine(9) yellow LEDs and Nine (9) red LEDs in a 3 x 3 Grid pattern. (there were NOT enough Green LEDs in the kit).

(2) Integrate the IR reciever software as provided to control the game.

(3) Design a GUI to control the game from the (host) computer mouse.

(4) Design a control program to recieve the inputs and maintain the state of the game and determine which player has won or if a draw has been reached.

(5) When a winner has been determined:

(a) Light up the winners Nine (9) LEDs to show that they are the winner.

(b) Have the victory tune played, the on the buzzer or on the speaker.

c. Extra: Have the GUI record the number of games played and the results.

d. Extra Tasks determined by the Team:

(1) Have a start button, which will sound a starting tune.

(2) Have a method of changing the order of the starting colour. Denoted by lighting all Nine (9) LEDs of the starting player. When pressed twice this will also test the functionality of the Eighteen (18) LEDs.

(3) Have a method of indicating to the selecting player, that his selection is invalid.

(4) Have a method to reset a game in progress (similar to task in Para 2. d. (1)).

e. Limitations:

(1) The late announcement of the aim of the laboratories, i.e. the early announcment of the requirements of Assignments One and Two would have provided impetus to learning the lessons from the early labs.

(2) Equipmetn Limitations:

(a) The number of ports available on the processor.

(b) The voltage output of the processor.

(c) The lack of gates to allow us to build our own electronic circuit.

(d) The similarity of the ten thousand Ohm (10,000 Ω) resistors to the two hundred and twenty Ohm (220 Ω) resistors. (I would put

a dab of coloured paint on the 10,000 Ω Resistors).

(3) The small number of Multimeters available.

(4) Personal Limitations:

(a) Lack of knowledge of the C language.

(b) Lack of knowledge of Music.

3. Implementation:

a. Plan completed as per Annex C.

b. Communication:

(1) Meetings in CS2018 lab. In pairs or as a team.

(2) https://github.com/JohnAODowd/Tictactoe

(3) Facebook – open chat – 3 contributers. On Stand alone, laptop or mobile. Normally NOT live, however this was used occasionally.

(4) Facebook – personal messaging – 1 to 1.

c. Execution:

(1) As per plan.

(2) Bones of Contention:

(a) Heraklius kept wanting to add more features, as he became more confident.

(b) My music code did not work in the required loop see Annex B Appendix 14, I altered this to produce working code for addition to the project, see Annex B Appendix 15.

(c) I dislike Git Hub.

(d) Heraklius had code working for the Uno on www.circuit.io, which showed him how the I/O system worked. I made a circuit diagram on Herakliusz's account now at

https://circuits.io/circuits/3251441-the-unnamed-circuit, which was deleted by accident. He later. didn't like the idea of the emulator, as it wouldn't fully represent all aspects of the program correctly. He was right. I considered it more important not to fry the board, as I wanted it to function for the evaluation. He wanted everything to work smoothly and was confident in the performance of the circuit and his ability, not to damage the hardware.

(e) Some reports (including mine) were not completed in a timely fashion.

4. Evaluation: The lecturer and staff evaluated the assignment and deemed it worthy of being demonstrated to the class.

5. Conclusions:

a. C Programming code:

(1) Complete Code: See John O'Dowd - Assignment 2 - 28th November - Para 5 Appendix Para 5.1 Complete Code.

(2) My Music Code:

(a) The timings of the code will vary from machine to machine. This could be fixed by using a test to evaluate the speed of processing on the machine and building in a variable in to solve this on a case by case basis, building it into the loop would be a nasty way to do it, however if you did, you could ensure the music played correctly no matter what other processing was going on in the background on the machine.

(b) If I had managed to build the array system attempted at Annex B – Appendix 12, it would probably have emulated the first attempts to digitise music. If I had had time to read more about the origins of music files, I might have seen a better solution to my notes and timings problem.

b. Circuit:

(1) For discussions about the circuit see Annex C.

(2) For comments on the building of the circuit see Annex D.

(3) Nice circuit. It was fun building it.

c. Files:

(1) I should have put more information in the filenames. Probably as "sketch_ColmanOKEEFFE_114712191_2016_11_23a_MusicVer03.ino", it makes the file unwieldy, however it is easily identifiable. or "sketch_ccok1_2016_11_23_1600hrs_MusicVer03.ino".

d. Hardware emulation:

(1) www.circuit.io / www.circuits.io We should have saved screenshots. As we did during Assignment 01, however I was building it and enjoying it and forgot to document the build. Also we should have used student emails, "disqus!", now has my Facebook™ details, as that is how one method how people log in to talk to one another about their projects.

e. Report:

(1) We should have been completing lab reports in an Assignment Science Lab book (One book for our team) kept in one of the lockers. This book should also have held the list of diagrams, photos and files created on the assignement and dated See Annex G. I know GitHub can do this, I don't like GitHub.

(2) I put the plan at Annex C. I know this should have gone in earlier, probably as Annex B, however I had written the titles and I was not able in sufficient time to rename all the Appendices. Apache Open Office. I must build and share a document template to do this in the future.

List of Annexes and Appendices:

Annex F – The evolution of the Tic-tac-toe Manual

Annex G – Sample Lab report

Annex A - C language Programming Code for creating Music

I used the following code at Annex A Appendices 1, 2, 3 for study purposes in order to understand how the arduino system could be used to produce music. After studying the code I used portions of it in order to write my music playing program. All variable names were changed and some of the functions were altered. In order to see what the timing effects were, at the beginning I kept the values so that I would not overload the system by putting in guesses. I then altered the music until I found it pleasing or it conformed to the music I was trying to emulate (Greensleeves – traditional).

I am not able to write the music diagrams to discribe what I have written, so I will not do that here.

Annex A – Appendix 01 – Play Melody Tutorial

```
/* Play Melody - Code for study only
 * https://www.arduino.cc/en/Tutorial/PlayMelody
 * Tutorial on Arduino site.
 * -----------
 *
 * Program to play a simple melody
 *
 * Tones are created by quickly pulsing a speaker on and off
 *   using PWM, to create signature frequencies.
 *
 * Each note has a frequency, created by varying the period of
 *  vibration, measured in microseconds. We'll use pulse-width
 *  modulation (PWM) to create that vibration.
 * We calculate the pulse-width to be half the period; we pulse
 *  the speaker HIGH for 'pulse-width' microseconds, then LOW
 *  for 'pulse-width' microseconds.
 *  This pulsing creates a vibration of the desired frequency.
 *
 * (cleft) 2005 D. Cuartielles for K3
 * Refactoring and comments 2006 clay.shirky@nyu.edu
 * See NOTES in comments at end for possible improvements
 */
// TONES  ==========================================
// Start by defining the relationship between
//       note, period, &  frequency.
#define  c     3830    // 261 Hz
#define  d     3400    // 294 Hz
#define  e     3038    // 329 Hz
#define  f     2864    // 349 Hz
#define  g     2550    // 392 Hz
#define  a     2272    // 440 Hz
#define  b     2028    // 493 Hz
#define  C     1912    // 523 Hz
// Define a special note, 'R', to represent a rest
#define  R     0
// SETUP ============================================
// Set up speaker on a PWM pin (digital 9, 10 or 11)
int speakerOut = 9;
// Do we want debugging on serial out? 1 for yes, 0 for no
int DEBUG = 1;
void setup() {
  pinMode(speakerOut, OUTPUT);
  if (DEBUG) {
    Serial.begin(9600); // Set serial out if we want debugging
  }
}
// MELODY and TIMING  =======================================
```

```
//  melody[] is an array of notes, accompanied by beats[],
//  which sets each note's relative length (higher #, longer note)
int melody[] = {  C,  b,  g,  C,  b,   e,  R,  C,  c,  g, a, C };
int beats[]  = { 16, 16, 16,  8,  8,  16, 32, 16, 16, 16, 8, 8 };
int MAX_COUNT = sizeof(melody) / 2; // Melody length, for looping.
// Set overall tempo
long tempo = 10000;
// Set length of pause between notes
int pause = 1000;
// Loop variable to increase Rest length
int rest_count = 100; //<-BLETCHEROUS HACK; See NOTES
// Initialize core variables
int tone_ = 0;
int beat = 0;
long duration  = 0;
// PLAY TONE  ==============================================
// Pulse the speaker to play a tone for a particular duration
void playTone() {
  long elapsed_time = 0;
  if (tone_ > 0) { // if this isn't a Rest beat, while the tone has
    //  played less long than 'duration', pulse speaker HIGH and LOW
    while (elapsed_time < duration) {
      digitalWrite(speakerOut,HIGH);
      delayMicroseconds(tone_ / 2);
      // DOWN
      digitalWrite(speakerOut, LOW);
      delayMicroseconds(tone_ / 2);
      // Keep track of how long we pulsed
      elapsed_time += (tone_);
    }
  }
  else { // Rest beat; loop times delay
    for (int j = 0; j < rest_count; j++) { // See NOTE on rest_count
      delayMicroseconds(duration);
    }
  }
}

// LET THE WILD RUMPUS BEGIN ================================
void loop() {
  // Set up a counter to pull from melody[] and beats[]
  for (int i=0; i<MAX_COUNT; i++) {
    tone_ = melody[i];
    beat = beats[i];
    duration = beat * tempo; // Set up timing
    playTone();
    // A pause between notes...
    delayMicroseconds(pause);
    if (DEBUG) { // If debugging, report loop, tone, beat, and duration
```

```
      Serial.print(i);
      Serial.print(":");
      Serial.print(beat);
      Serial.print(" ");
      Serial.print(tone_);
      Serial.print(" ");
      Serial.println(duration);
    }
  }
}
/*
 * NOTES
 * The program purports to hold a tone for 'duration' microseconds.
 *  Lies lies lies! It holds for at least 'duration' microseconds, _plus_
 *  any overhead created by incremeting elapsed_time (could be in excess of
 *  3K microseconds) _plus_ overhead of looping and two digitalWrites()
 *
 * As a result, a tone of 'duration' plays much more slowly than a rest
 *  of 'duration.' rest_count creates a loop variable to bring 'rest' beats
 *  in line with 'tone' beats of the same length.
 *
 * rest_count will be affected by chip architecture and speed, as well as
 *  overhead from any program mods. Past behavior is no guarantee of future
 *  performance. Your mileage may vary. Light fuse and get away.
 *
 * This could use a number of enhancements:
 * ADD code to let the programmer specify how many times the melody should
 *     loop before stopping
 * ADD another octave
 * MOVE tempo, pause, and rest_count to #define statements
 * RE-WRITE to include volume, using analogWrite, as with the second program at
 *         http://www.arduino.cc/en/Tutorial/PlayMelody
 * ADD code to make the tempo settable by pot or other input device
 * ADD code to take tempo or volume settable by serial communication
 *         (Requires 0005 or higher.)
 * ADD code to create a tone offset (higer or lower) through pot etc
 * REPLACE random melody with opening bars to 'Smoke on the Water'
 */
```

Annex A - Appendix 02 - Play Melody with Volume control – Source unknown.

```
/* Play Melody - with Volume control for study only
 * -----------
 *
 * Program to play melodies stored in an array, it requires to know
 * about timing issues and about how to play tones.
 *
 * The calculation of the tones is made following the mathematical
 * operation:
 *
 *       timeHigh = 1/(2 * toneFrequency) = period / 2
 *
 * where the different tones are described as in the table:
 *
 * note         frequency       period  PW (timeHigh)
 * c            261 Hz          3830    1915
 * d            294 Hz          3400    1700
 * e            329 Hz          3038    1519
 * f            349 Hz          2864    1432
 * g            392 Hz          2550    1275
 * a            440 Hz          2272    1136
 * b            493 Hz          2028    1014
 * C            523 Hz          1912    956
 *
 * (cleft) 2005 D. Cuartielles for K3
 */
int ledPin = 13;
int speakerOut = 9;
byte names[] = {'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'};
int tones[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
byte melody[] = "2d2a1f2c2d2a2d2c2f2d2a2c2d2a1f2c2d2a2a2g2p8p8p8p";
// count length: 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
//                               10                  20                  30
int count = 0;
int count2 = 0;
int count3 = 0;
int MAX_COUNT = 24;
int statePin = LOW;
void setup() {
 pinMode(ledPin, OUTPUT);
}
void loop() {
  analogWrite(speakerOut, 0);
  for (count = 0; count < MAX_COUNT; count++) {
    statePin = !statePin;
    digitalWrite(ledPin, statePin);
    for (count3 = 0; count3 <= (melody[count*2] - 48) * 30; count3++) {
      for (count2=0;count2<8;count2++) {
```

```
        if (names[count2] == melody[count*2 + 1]) {
          analogWrite(speakerOut,500);
          delayMicroseconds(tones[count2]);
          analogWrite(speakerOut, 0);
          delayMicroseconds(tones[count2]);
        }
        if (melody[count*2 + 1] == 'p') {
          // make a pause of a certain size
          analogWrite(speakerOut, 0);
          delayMicroseconds(500);
        }
      }
    }
  }
}
```

Annex A - Appendix 03 - NOT WORKING CODE from www.stackoverflow.com

```
//C program MP3 Player NOT working code for study only
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * pFile;
#define ARRAYSIZE 100
#define SONGNAME 25
#define SONGARTIST 25
#define SONGLENGTH 25
#define TRACK_NUMBER
void func1(void);
void func2(void);
void func3(void);
void func4(void);
void delete_song(void);
void func6(void);
typedef struct Mp3rec
{
       char name[SONGNAME];
       char artist[SONGARTIST];
       char length[SONGLENGTH];
       char tname[TRACK_NUMBER];
};
int main (void)
{
   int menuchoice=0;
   //fp= fopen("mp3_list.txt", "r" "W" "a"); either to add or read a text file.
   do
   {
    printf ("*********************************************\n");
    printf ("*************************************\n");
    printf ("********** MP3 Player **********\n");
    printf ("*********************************************");
    printf("\n1: Play File");
    printf("\n2: Show Library");
    printf("\n3: Add Record");
    printf("\n4: Edit/Modify ");
    printf("\n5: Delete Record");
    printf("\n6: Quit");
     printf("\n\nEnter Choice From 1-6: ");
    scanf("&#37;d", &menuchoice);
    switch (menuchoice)
    {
          case 1:
                func1();
                break;
          case 2:
```

```
            func2();
            break;
        case 3:
            func3();
            break;
        case 4:
            func4();
            break;
         case 5:delete_song();
            break;
         case 6:
            break;
        default:
             printf("\nInvalid Choice: 1-4 Only Please");
             }

             }while(menuchoice!=4);
             system("pause");
}
void func1(void)
{
    printf("\n\n Play File\n\n");
    printf ("*********************************************\n");
    printf ("************************************\n");
    printf ("********** **********\n");
    printf ("*********************************************");
    {
   char commandarray[ARRAYSIZE];
   char mp3filename[ARRAYSIZE] ="C:\\mplayer\\Robin S - Show me Love.mp3";
   sprintf(commandarray, "C:\\mplayer\\mplayer.exe \"%s\"", mp3filename);
   printf("\nAttempting to Run Command \"%s\"...\n\n", commandarray);
   system(commandarray);
   int menuchoice=0;
   switch (menuchoice)
   printf("\n1: Back To Menu");
   printf("\n5: Quit");
   printf("\nDone.\n\n");
   system("pause");
}
}
void func2(void)
{
    printf("\n\n Show Library\n\n");
}
void func3(void)
{
    printf("\n\n Add Record/Song\n\n");
    fopen("ListofSongs.txt","r+");
    printf("\n\n Add Record\n\n");
```

```
}
void func4(void)
{
    printf("\n\n Edit/Modify the Current Song\n\n");
}
void delete_song(void)
{
    printf("\n\n Delete File\n\n");
 int i;
 int track_number;
    printf("Please Enter the track number that you wish to delete:");
    fflush(stdin);
    scanf("%d", &track_number);
}
    //for( Track_Number = Track_Number ; Track_Number <=Total Songs ; Track_Number ++)
void func6(void)
{
   // break;
}
```

Annex B – C Programming code to create my Music

Requirement – the program should play a Victory tune, if either player wins.

I read the three code sections listed at Annex A Appendices 01, 02, 03.
I understood the code and how it functioned, not why it functioned.
I adapted it and created my own variable names and loops.
I looked up the correct frequencies (and made the necessary calculations) for the notes on a physics site – See Annex B Appendix A comments.
I started by writing a scale in Middle C.
When this was done I started to write music.

I was going to put all my comments on how I created the music here, however I put them in as comments through out all the Appendices. It was fun, however it was a little outside the scope of the assignment, which only required a Victory tune to be played.

Annex B – Appendix 01 – Scetch_nov23a_ScaleVer2 (filenames).
My apologies it was at this point I realised I should have saved Version 1, however I did continue to continually save changes after this.

```
// Colman O'Keeffe 114712191 23 Nov 2016
// note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
}
// Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
// tempo
long tempo = 10000;
// length of pause
int pause  = 1000;
// add to loop to pause for 1 sec
int rest_count = 100;
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;
// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
```

```
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Try to play the entire scale
void loop(){
  for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
    note = colmans_tune[timer];
    beat = colmans_beats[timer];
    duration = beat * tempo;
    play_note();
    delayMicroseconds(pause);
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
  }
}
```

Annex B – Appendix 02 - Scetch_nov23a_ScaleVer2a
This version was successful and I found the notes pleasing so I moved on to creating music.

```
// Colman O'Keeffe 114712191 23 Nov 2016
// note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
}
// Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
// tempo
long tempo = 10000;
// length of pause
int pause  = 5000; // original pause increased from 1000 to 5000
// add to loop to pause for 1 sec
int rest_count = 100;
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;
// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
```

```
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Try to play the entire scale
void loop(){
  for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
    note = colmans_tune[timer];
    beat = colmans_beats[timer];
    duration = beat * tempo;
    play_note();
    delayMicroseconds(pause);
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
  }
}
```

Annex B – Appendix 03 - Scetch_nov23a_ScaleVer2b

At this point I should have realised Ver 2a was the final version of the scale, however I didn't and wrote my first test tune and saved it incorrectly.

```
// Colman O'Keeffe 114712191 23 Nov 2016
// note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {16, 8, 16, 8, 24, 16, 24, 8};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
// tempo
long tempo = 10000;
// length of pause
int pause  = 5000; // original pause increased from 1000 to 5000
// add to loop to pause for 1 sec
int rest_count = 100;
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;
// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
```

```
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Try to play the entire scale
void loop(){
  for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
    note = colmans_tune[timer];
    beat = colmans_beats[timer];
    duration = beat * tempo;
    play_note();
    delayMicroseconds(pause);
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
  }
}
```

Annex B – Appendix 04 - Scetch_nov23a_ColmansTuneVer01

At this point I started changing code and adding comments, I know, I should have been doing this earlier, it is my first real program and I understood how it was working, a code reader would too. I realised soon after that I should be saying when I made changes, I did that too. I did not delete previously used code snippets, I commented out older code so that I would have it available if needed.

```
// Colman O'Keeffe 114712191 23 Nov 2016
// note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {16, 8, 16, 8, 24, 16, 24, 8};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
// tempo
long tempo = 10000;
// length of pause
int pause  = 5000; // original pause increased from 1000 to 5000
// add to loop to pause for 1 sec
int rest_count = 100;
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;
// try to play a note
void play_note(){
```

```
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Try to play the entire scale
void loop(){
  for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
    note = colmans_tune[timer];
    beat = colmans_beats[timer];
    duration = beat * tempo;
    play_note();
    delayMicroseconds(pause);
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
  }
}
```

Annex B – Appendix 05 - Scetch_nov23a_ColmansTuneVer01a

Note: I never used the debugging as I thought it might hang the system. However, I thought that later, it might be useful, so I kept it in.

```
// Colman O'Keeffe 114712191 23 Nov 2016
// note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {12, 18, 14, 10, 22, 16, 30, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
// tempo
long tempo = 10000;
// length of pause
int pause  = 5000; // original pause increased from 1000 to 5000
// add to loop to pause for 1 sec
int rest_count = 100;
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;
// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
```

```
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}

// Try to play the entire scale
void loop(){
  for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
    note = colmans_tune[timer];
    beat = colmans_beats[timer];
    duration = beat * tempo;
    play_note();
    delayMicroseconds(pause);
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
  }
}
```

Annex B – Appendix 06 - Scetch_nov23a_ColmansTuneVer02

```
/* Colman O'Keeffe 114712191 23 Nov 2016
* note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
Ver 01 period = 1/f in milli secs all values calculated manually on calculator
Ver 02 all periods altered randomly to ensure a know tune is NOT used
*/
# define Col_c4 3800 // approx 261.63 Hz
# define Col_d4 3400 // approx 293.66 Hz
# define Col_e4 3000 // approx 329.63 Hz
# define Col_f4 2800 // approx 349.23 Hz
# define Col_g4 2500 // approx 392.00 Hz
# define Col_a4 2200 // approx 440.00 Hz
# define Col_b4 2000 // approx 493.88 Hz
# define Col_c5 1900 // approx 523.25 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {30, 16, 20, 10, 40, 20, 60, 20};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
// tempo
long tempo = 10000;
// length of pause
int pause  = 5000; // original pause increased from 1000 to 5000
// add to loop to pause for 1 sec
int rest_count = 100;
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;
// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
```

```
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Try to play the entire scale
void loop(){
  for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
    note = colmans_tune[timer];
    beat = colmans_beats[timer];
    duration = beat * tempo;
    play_note();
    delayMicroseconds(pause);
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
  }
}
```

Annex B – Appendix 07 - Scetch_nov23a_ColmansTuneVer02a

```
/* Colman O'Keeffe 114712191 23 Nov 2016
* note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
Ver 01 period = 1/f in milli secs all values calculated manually on calculator
Ver 02 all periods altered randomly to ensure a know tune is NOT used
the differance in the change in tones is not noticable to me as a layman.
*/
# define Col_c4 3800 // approx 261.63 Hz
# define Col_d4 3400 // approx 293.66 Hz
# define Col_e4 3000 // approx 329.63 Hz
# define Col_f4 2800 // approx 349.23 Hz
# define Col_g4 2500 // approx 392.00 Hz
# define Col_a4 2200 // approx 440.00 Hz
# define Col_b4 2000 // approx 493.88 Hz
# define Col_c5 1900 // approx 523.25 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
// Ver 1 change scale notes in random order altered length of beats in random order
// Ver 2a doubled length of notes in tune added random beats
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4, Col_c4,
Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {30, 16, 20, 10, 40, 20, 60, 20, 40, 10, 40, 20, 30, 10, 60, 20};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
// tempo
long tempo = 10000;
// length of pause
int pause  = 5000; // original pause increased from 1000 to 5000
// add to loop to pause for 1 sec
int rest_count = 100;
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;
// try to play a note
```

```
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Try to play the entire scale   ----- NOT IN CODE          I should have altered this after I saw it
                                                            was working.
void loop(){
  for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
    note = colmans_tune[timer];
    beat = colmans_beats[timer];
    duration = beat * tempo;
    play_note();
    delayMicroseconds(pause);
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
  }
}
```

Annex B – Appendix 08 - Scetch_nov23a_ColmansTuneVer03

An attempt to emulate Greensleeves – traditional.
The college could copyright this version, if you could get someone to invent a method to write musical notation for digital timings (non standard note lengths!).

```
/* Colman O'Keeffe 114712191 23 Nov 2016
* note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
Ver 01 period = 1/f in milli secs all values calculated manually on calculator
Ver 02 all periods altered randomly to ensure a know tune is NOT used
the differance in the change in tones is not noticable to me as a layman.
Ver 3 returned note values to original values added A4 minor, B4 minor, E4 Minor & B5
*/
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4m 3214 // 311.13 Hz
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4m 2407 // 415.30 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4m 2025 // 493.88 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define Col_b5 1012 // 987.77 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
// Ver 1 change scale notes in random order altered length of beats in random order
// Ver 2a doubled length of notes in tune added random beats
/*
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4, Col_c4,
Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {30, 16, 20, 10, 40, 20, 60, 20, 40, 10, 40, 20, 30, 10, 60, 20};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
```

```
/* Ver 3 attempt at Greensleeves
* chords from https://www.acousticmusicarchive.com/greensleeves-chords-lyrics
*/
int colmans_tune[] = {Col_g4, Col_d4, Col_b4m, Col_a4m, Col_b4, Col_g4, Col_d4, Col_b4m,
Col_c4, Col_b5, Col_e4m};
int colmans_beats[] = {60, 20, 20, 60, 40, 60, 20, 40, 40, 10, 30};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
// tempo
long tempo = 30000;  // Ver3 for Greensleeves original tempo lengthened from 10000 to 30000
// length of pause
int pause  = 5000; // original pause increased from 1000 to 5000
// add to loop to pause for 1 sec
int rest_count = 100;
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;
// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Try to play the entire scale
void loop(){
  for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
    note = colmans_tune[timer];
    beat = colmans_beats[timer];
    duration = beat * tempo;
    play_note();
    delayMicroseconds(pause);
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
```

```
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
  }
}
```

Annex B – Appendix 09 - Scetch_nov23a_HornpipeVer01

It is really hard to find uncopyrighted music. Most are are adaptations of old tunes, with copyright attached. I hope I found some that are not.
I was silly here and wrote the different versions all in together, bad idea, however I ws having problems losing files due to overloading my account, so you'll forgive me this once, I hope.
Here is a hornpipe of my own.

```
/* Colman O'Keeffe 114712191 23 Nov 2016
* note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
Ver 01 period = 1/f in milli secs all values calculated manually on calculator
Ver 02 all periods altered randomly to ensure a know tune is NOT used
the differance in the change in tones is not noticable to me as a layman.
Ver 3 returned note values to original values added A4 minor, B4 minor, E4 Minor & B5
Hornpipe Ver01 added A5
*/
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4m 3214 // 311.13 Hz
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4m 2407 // 415.30 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4m 2025 // 493.88 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define Col_a5 1136 // 880.00 Hz
# define Col_b5 1012 // 987.77 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
// Ver 1 change scale notes in random order altered length of beats in random order
// Ver 2a doubled length of notes in tune added random beats
/*
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4, Col_c4,
```

```
Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {30, 16, 20, 10, 40, 20, 60, 20, 40, 10, 40, 20, 30, 10, 60, 20};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
/* Ver 3 attempt at Greensleeves
* chords from https://www.acousticmusicarchive.com/greensleeves-chords-lyrics
int colmans_tune[] = {Col_g4, Col_d4, Col_b4m, Col_a4m, Col_b4, Col_g4, Col_d4, Col_b4m,
Col_c4, Col_b5, Col_e4m};
int colmans_beats[] = {60, 20, 20, 60, 40, 60, 20, 40, 40, 10, 30};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
int hornpipe_tune[] = {Col_g4, Col_a5, Col_d4, Col_g4, Col_c4, Col_d4, Col_g4, Col_d4, Col_g4,
Col_g4, Col_c4, Col_a5, Col_d4,  Col_g4, Col_c4, Col_d4, Col_g4};
int hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(hornpipe_tune) / 2; // Tune length i.o.t. loop
// tempo
long tempo = 30000;  // Ver3 for Greensleeves original tempo lengthened from 10000 to 30000
// length of pause
int pause  = 5000; // original pause increased from 1000 to 5000
// add to loop to pause for 1 sec
int rest_count = 100;
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;
// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Try to play the entire scale
void loop(){
  for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
```

```
    note = hornpipe_tune[timer];
    beat = hornpipe_beats[timer];
    duration = beat * tempo;
    play_note();
    delayMicroseconds(pause);
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
  }
}
```

Annex B – Appendix 10 - Scetch_nov23a_Hornpipe_for_LisaVer01

At this point I decided to write a Hornpipe for Lisa who was sitting next to me, I should have forked here and left the code unpublished, however I didn't and it was in the following code so I have to document where it came from and the College owns it anyway, so here it is. IN this version it's a little sad, however the buzzer & speaker doesnt like high notes and I would have put it up an octave. Some day I might have a go at finishing it.

In order to have a fixed length, I had to write code to stop the program after a few repetitions.
I also commented out the silly debugging stuff as it was annoying to see it there all the time. It alters the timing of the program slightly.

```
/* Colman O'Keeffe 114712191 23 Nov 2016
* note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
Ver 01 period = 1/f in milli secs all values calculated manually on calculator
Ver 02 all periods altered randomly to ensure a know tune is NOT used
the differance in the change in tones is not noticable to me as a layman.
Ver 3 returned note values to original values added A4 minor, B4 minor, E4 Minor & B5
Hornpipe Ver01 added A5
*/
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4m 3214 // 311.13 Hz
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4m 2407 // 415.30 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4m 2025 // 493.88 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define Col_a5 1136 // 880.00 Hz
# define Col_b5 1012 // 987.77 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
/* debugging removed for Hornpipe for Lisa
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
*/
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
```

```
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
// Ver 1 change scale notes in random order altered length of beats in random order
// Ver 2a doubled length of notes in tune added random beats
/*
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4, Col_c4,
Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {30, 16, 20, 10, 40, 20, 60, 20, 40, 10, 40, 20, 30, 10, 60, 20};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
/* Ver 3 attempt at Greensleeves
* chords from https://www.acousticmusicarchive.com/greensleeves-chords-lyrics
int colmans_tune[] = {Col_g4, Col_d4, Col_b4m, Col_a4m, Col_b4, Col_g4, Col_d4, Col_b4m,
Col_c4, Col_b5, Col_e4m};
int colmans_beats[] = {60, 20, 20, 60, 40, 60, 20, 40, 40, 10, 30};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
/* HornpipeVer01 http://www.guitarnick.com/sailors-hornpipe-irish-flatpicking-guitar-score-
tab.html
int hornpipe_tune[] = {Col_g4, Col_a5, Col_d4, Col_g4, Col_c4, Col_d4, Col_g4, Col_d4, Col_g4,
Col_g4, Col_c4, Col_a5, Col_d4,  Col_g4, Col_c4, Col_d4, Col_g4};
int hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(hornpipe_tune) / 2; // Tune length i.o.t. loop
*/
/* Colmans_hornpipe Ver01
int Colmans_hornpipe_tune[] = {Col_e4, Col_c5, Col_f4, Col_d4, Col_c4, Col_a4m, Col_b4m,
Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_d4, Col_c4, Col_a4m, Col_b4m};
int Colmans_hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(Colmans_hornpipe_tune) / 2; // Tune length i.o.t. loop
*/
// Hornpipe for Lisa - replaced Chord02 Col_c5 with Col_g4 - replaced Chord04 Col_d4 with
Col_b4 - replaced Chord14 with Col_g4 with Col_f4
int Hornpipe_for_Lisa_tune[] = {Col_e4, Col_g4, Col_f4, Col_b4, Col_c4, Col_a4m, Col_b4m,
Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_f4, Col_c4, Col_a4m, Col_b4m};
int Hornpipe_for_Lisa_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(Hornpipe_for_Lisa_tune) / 2; // Tune length i.o.t. loop
// Hornpipe for Lisa added NO_OF_REPETETIONS No - 5
int NO_OF_REPETITIONS = 5;
// tempo
long tempo = 35000;  // Ver3 for Greensleeves original tempo lengthened from 10000 to 30000
                     // Hornpipe for Lisa tempo lenghtened from 30000 to 35000
// length of pause
int pause  = 10000; // original pause increased from 1000 to 5000
                    // Hornpipe for Lisa pause increased from 5000 to 10000
// add to loop to pause for 1 sec
int rest_count = 200;
                    // Hornpipe for Lisa rest_count increased from 100 to 200
// setup variables for main function
int note = 0;
```

```
int beat = 0;
long duration = 0;
// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Colmans Scale - Try to play the entire scale
// Hornpipe for Lisa added next line
int reps = 0;
void loop(){
  if (reps < NO_OF_REPETITIONS){
      for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
        note = Hornpipe_for_Lisa_tune[timer];
        beat = Hornpipe_for_Lisa_beats[timer];
        duration = beat * tempo;
        play_note();
        delayMicroseconds(pause);
      }
      reps ++;
  }
   else{
      digitalWrite(buzzer_out, LOW);
/* Debugging removed for Hornpipe for Lisa
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
```

```
    }
*/
  }
}
```

Annex B – Appendix 11 - Scetch_nov23a_StartingBuzzerVer01

At this point, knowing I was capable, I asked the rest of the team what they wanted, as I already had a few tunes, they only asked for a starting buzzer. My intention, which I shared with the team was for a starting buzzer, greensleeves for a draw and colmans tune for a win.
I asked them not to include hornpipe for Lisa, however I was silly enough to leave the code in and I have to go now and see what the final version looked like, they were amalgamating the project and the code was written as part of the brief, so... Heres' hoping.

```
/* Colman O'Keeffe 114712191 23 Nov 2016
* note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
Ver 01 period = 1/f in milli secs all values calculated manually on calculator
Ver 02 all periods altered randomly to ensure a know tune is NOT used
the differance in the change in tones is not noticable to me as a layman.
Ver 3 returned note values to original values added A4 minor, B4 minor, E4 Minor & B5
Hornpipe Ver01 added A5
*/
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4m 3214 // 311.13 Hz
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4m 2407 // 415.30 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4m 2025 // 493.88 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define Col_a5 1136 // 880.00 Hz
# define Col_b5 1012 // 987.77 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
/* debugging removed for Hornpipe for Lisa
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
*/
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
// Ver 1 change scale notes in random order altered length of beats in random order
```

```
// Ver 2a doubled length of notes in tune added random beats
/*
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4, Col_c4,
Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {30, 16, 20, 10, 40, 20, 60, 20, 40, 10, 40, 20, 30, 10, 60, 20};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
/* Ver 3 attempt at Greensleeves
* chords from https://www.acousticmusicarchive.com/greensleeves-chords-lyrics
int colmans_tune[] = {Col_g4, Col_d4, Col_b4m, Col_a4m, Col_b4, Col_g4, Col_d4, Col_b4m,
Col_c4, Col_b5, Col_e4m};
int colmans_beats[] = {60, 20, 20, 60, 40, 60, 20, 40, 40, 10, 30};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
/* HornpipeVer01 http://www.guitarnick.com/sailors-hornpipe-irish-flatpicking-guitar-score-
tab.html
int hornpipe_tune[] = {Col_g4, Col_a5, Col_d4, Col_g4, Col_c4, Col_d4, Col_g4, Col_d4, Col_g4,
Col_g4, Col_c4, Col_a5, Col_d4,  Col_g4, Col_c4, Col_d4, Col_g4};
int hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(hornpipe_tune) / 2; // Tune length i.o.t. loop
*/
/* Colmans_hornpipe Ver01
int Colmans_hornpipe_tune[] = {Col_e4, Col_c5, Col_f4, Col_d4, Col_c4, Col_a4m, Col_b4m,
Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_d4, Col_c4, Col_a4m, Col_b4m};
int Colmans_hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(Colmans_hornpipe_tune) / 2; // Tune length i.o.t. loop
*/
/* Hornpipe for Lisa - replaced Chord02 Col_c5 with Col_g4 - replaced Chord04 Col_d4 with
Col_b4 - replaced Chord14 with Col_g4 with Col_f4
int Hornpipe_for_Lisa_tune[] = {Col_e4, Col_g4, Col_f4, Col_b4, Col_c4, Col_a4m, Col_b4m,
Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_f4, Col_c4, Col_a4m, Col_b4m};
int Hornpipe_for_Lisa_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(Hornpipe_for_Lisa_tune) / 2; // Tune length i.o.t. loop
Hornpipe for Lisa added NO_OF_REPETETIONS No - 5
*/
int StartingBuzzer_tune[] = {Col_c5, Col_b4m, Col_a4m, Col_e4, Col_c4};
int StartingBuzzer_beats[] = {20, 20, 20, 20, 20};
int MAXIMUM_COUNT = sizeof(StartingBuzzer_tune) / 2; // Tune length i.o.t. loop
// Starting Buzzer changed No of repetitions to 01
int NO_OF_REPETITIONS = 1;
// tempo
long tempo = 35000;  // Ver3 for Greensleeves original tempo lengthened from 10000 to 30000
                     // Hornpipe for Lisa tempo lenghtened from 30000 to 35000
// length of pause
int pause  = 10000; // original pause increased from 1000 to 5000
                    // Hornpipe for Lisa pause increased from 5000 to 10000
// add to loop to pause for 1 sec

int rest_count = 200;
```

```
                // Hornpipe for Lisa rest_count increased from 100 to 200
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;

// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Colmans Scale - Try to play the entire scale
// Hornpipe for Lisa added next line
int reps = 0;
void loop(){
  if (reps < NO_OF_REPETITIONS){
      for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
        note = StartingBuzzer_tune[timer];
        beat = StartingBuzzer_beats[timer];
        duration = beat * tempo;
        play_note();
        delayMicroseconds(pause);
      }
      reps ++;
  }
   else{
      digitalWrite(buzzer_out, LOW);
/* Debugging removed for Hornpipe for Lisa
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
```

```
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
*/
  }
}
```

Annex B – Appendix 12 - Scetch_nov23a_StartingBuzzerVer02 - NON WORKING CODE

This was an attempt to read the tunes separately and turn them into arrays of arrays, it failed and this is what is left, given time I could probable do it. No fun in it though and it changes the timings.

```
/* Colman O'Keeffe 114712191 23 Nov 2016
* note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
Ver 01 period = 1/f in milli secs all values calculated manually on calculator
Ver 02 all periods altered randomly to ensure a know tune is NOT used
the differance in the change in tones is not noticable to me as a layman.
Ver 3 returned note values to original values added A4 minor, B4 minor, E4 Minor & B5
Hornpipe Ver01 added A5
*/
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4m 3214 // 311.13 Hz
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4m 2407 // 415.30 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4m 2025 // 493.88 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define Col_a5 1136 // 880.00 Hz
# define Col_b5 1012 // 987.77 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
/* debugging removed for Hornpipe for Lisa
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
*/
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
// Ver 1 change scale notes in random order altered length of beats in random order
// Ver 2a doubled length of notes in tune added random beats
/*
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4, Col_c4,
Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
```

```
int colmans_beats[] = {30, 16, 20, 10, 40, 20, 60, 20, 40, 10, 40, 20, 30, 10, 60, 20};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
/* Ver 3 attempt at Greensleeves
* chords from https://www.acousticmusicarchive.com/greensleeves-chords-lyrics
int colmans_tune[] = {Col_g4, Col_d4, Col_b4m, Col_a4m, Col_b4, Col_g4, Col_d4, Col_b4m,
Col_c4, Col_b5, Col_e4m};
int colmans_beats[] = {60, 20, 20, 60, 40, 60, 20, 40, 40, 10, 30};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
/* HornpipeVer01 http://www.guitarnick.com/sailors-hornpipe-irish-flatpicking-guitar-score-
tab.html
int hornpipe_tune[] = {Col_g4, Col_a5, Col_d4, Col_g4, Col_c4, Col_d4, Col_g4, Col_d4, Col_g4,
Col_g4, Col_c4, Col_a5, Col_d4,  Col_g4, Col_c4, Col_d4, Col_g4};
int hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(hornpipe_tune) / 2; // Tune length i.o.t. loop
*/
/* Colmans_hornpipe Ver01
int Colmans_hornpipe_tune[] = {Col_e4, Col_c5, Col_f4, Col_d4, Col_c4, Col_a4m, Col_b4m,
Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_d4, Col_c4, Col_a4m, Col_b4m};
int Colmans_hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(Colmans_hornpipe_tune) / 2; // Tune length i.o.t. loop
*/
/* Hornpipe for Lisa - replaced Chord02 Col_c5 with Col_g4 - replaced Chord04 Col_d4 with
Col_b4 - replaced Chord14 with Col_g4 with Col_f4
int Hornpipe_for_Lisa_tune[] = {Col_e4, Col_g4, Col_f4, Col_b4, Col_c4, Col_a4m, Col_b4m,
Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_f4, Col_c4, Col_a4m, Col_b4m};
int Hornpipe_for_Lisa_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(Hornpipe_for_Lisa_tune) / 2; // Tune length i.o.t. loop
Hornpipe for Lisa added NO_OF_REPETETIONS No - 5
*/
/*
int StartingBuzzer_tune[] = {Col_c5, Col_b4m, Col_a4m, Col_e4, Col_c4};
int StartingBuzzer_beats[] = {20, 20, 20, 20, 20};
// int MAXIMUM_COUNT = sizeof(StartingBuzzer_tune) / 2; // Tune length i.o.t. loop removed
when creating Music function
// Starting Buzzer changed No of repetitions to 01
int StartingBuzzer_No_of_repetitions[] = {1};
// tempo
int StartingBuzzer_tempo[] = {35000};  // Ver3 for Greensleeves original tempo lengthened from
10000 to 30000
                    // Hornpipe for Lisa tempo lenghtened from 30000 to 35000
// length of pause
int StartingBuzzer_pause[]  = {10000}; // original pause increased from 1000 to 5000
                   // Hornpipe for Lisa pause increased from 5000 to 10000
// add to loop to pause for 1 sec
int StartingBuzzer_rest_count[] = {200};
How to create an array with music information in it
StartingBuzzer[0] = sizeof(StartingBuzzer_tune)
```

```
StartingBuzzer[1] = StartingBuzzer_beats[]
StartingBuzzer[2] = StartingBuzzer_No_of_repetitions[]
StartingBuzzer[3] = StartingBuzzer_tempo[]
StartingBuzzer[4] = StartingBuzzer_pause[]
StartingBuzzer[5] = StartingBuzzer_rest_count[]}
StartingBuzzer[6-10] = tune
StartingBuzzer[11-15] = beats
StartingBuzzer[] ={5, 1, 35000, 10000, 200, Col_c5, 20, Col_b4m, 20, Col_a4m, 20, Col_e4, 20,
Col_c4, 20};
*/

// Hornpipe for Lisa rest_count increased from 100 to 200
// setup variables for main function

int note = 0;
int beat = 0;
long duration = 0;
int reps = 0;
int rest_count = 0;

// try to play a note

void Music(int Tune_Selection){
  int Music_data[15];
  if (Tune_Selection == 1){
    // Starting Buzzer
    int Music_data[] = {5, 1, 35000, 10000, 200, Col_c5, 20, Col_b4m, 20, Col_a4m, 20, Col_e4, 20,
Col_c4, 20};
  }
/*
How to create an array with music information in it
StartingBuzzer[0] = sizeof(StartingBuzzer_tune)
StartingBuzzer[1] = StartingBuzzer_beats[]
StartingBuzzer[2] = StartingBuzzer_No_of_repetitions[]
StartingBuzzer[3] = StartingBuzzer_tempo[]
StartingBuzzer[4] = StartingBuzzer_pause[]
StartingBuzzer[5] = StartingBuzzer_rest_count[]}
StartingBuzzer[6-10] = tune
StartingBuzzer[11-15] = beats
*/

    int No_of_repetitions = Music_data[2];
    long tempo = Music_data[3];
    int pause = Music_data[4];
    rest_count = Music_data[5];
    int Music_tune[Music_data[0]];
    int Music_beats[Music_data[0]];
    int music_index = Music_data[0];
    for (int index =0; index < Music_data[0]; index++){
```

```
      Music_tune[music_index] = Music_data[(5 + index)];
      Music_beats[music_index] = Music_data[(5 + index)];
    }
    int MAXIMUM_COUNT = sizeof(Music_tune) / 2;

  if (reps < No_of_repetitions){
      for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
        note = Music_tune[timer];
        beat = Music_beats[timer];
        duration = beat * tempo;
        play_note();
        delayMicroseconds(pause);
      }
      reps ++;
  }
    else{
      digitalWrite(buzzer_out, LOW);
/* Debugging removed for Hornpipe for Lisa
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
*/
  }
}
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
```

```
  }
}
// Colmans Scale - Try to play the entire scale
// Hornpipe for Lisa added next line

void loop(){
  Music(1);
}
```

Annex B – Appendix 13 - Scetch_nov23a_MusicVer01

At this point the team wanted short code for the amalgamated project, so I started to trim, I didn't think it possible to consolidate the code, so I left it and removed comments & unnecessaries. 3 versions. Version two was given to the team.

```
/* Colman O'Keeffe 114712191 23 Nov 2016
* note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
Ver 01 period = 1/f in milli secs all values calculated manually on calculator
Ver 02 all periods altered randomly to ensure a know tune is NOT used
the differance in the change in tones is not noticable to me as a layman.
Ver 3 returned note values to original values added A4 minor, B4 minor, E4 Minor & B5
Hornpipe Ver01 added A5
*/
# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4m 3214 // 311.13 Hz
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4m 2407 // 415.30 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4m 2025 // 493.88 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define Col_a5 1136 // 880.00 Hz
# define Col_b5 1012 // 987.77 Hz
# define rest 0
// buzzer
int buzzer_out = 12;
// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;
void setup(){
  pinMode(buzzer_out, OUTPUT);
/* debugging removed for Hornpipe for Lisa
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
*/
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
// Ver 1 change scale notes in random order altered length of beats in random order
// Ver 2a doubled length of notes in tune added random beats
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4, Col_c4,
Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
```

```
int colmans_beats[] = {30, 16, 20, 10, 40, 20, 60, 20, 40, 10, 40, 20, 30, 10, 60, 20};
int colmans_MAXIMUM_COUNT = sizeof(colmans_tune) / 2;
//int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
/* Ver 3 attempt at Greensleeves
* chords from https://www.acousticmusicarchive.com/greensleeves-chords-lyrics
int colmans_tune[] = {Col_g4, Col_d4, Col_b4m, Col_a4m, Col_b4, Col_g4, Col_d4, Col_b4m,
Col_c4, Col_b5, Col_e4m};
int colmans_beats[] = {60, 20, 20, 60, 40, 60, 20, 40, 40, 10, 30};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/
/* HornpipeVer01 http://www.guitarnick.com/sailors-hornpipe-irish-flatpicking-guitar-score-
tab.html
int hornpipe_tune[] = {Col_g4, Col_a5, Col_d4, Col_g4, Col_c4, Col_d4, Col_g4, Col_d4, Col_g4,
Col_g4, Col_c4, Col_a5, Col_d4,  Col_g4, Col_c4, Col_d4, Col_g4};
int hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(hornpipe_tune) / 2; // Tune length i.o.t. loop
*/
int Colmans_hornpipe_tune[] = {Col_e4, Col_c5, Col_f4, Col_d4, Col_c4, Col_a4m, Col_b4m,
Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_d4, Col_c4, Col_a4m, Col_b4m};
int Colmans_hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int Colmans_hornpipe_MAXIMUM_COUNT = sizeof(Colmans_hornpipe_tune) / 2;
/* Hornpipe for Lisa - replaced Chord02 Col_c5 with Col_g4 - replaced Chord04 Col_d4 with
Col_b4 - replaced Chord14 with Col_g4 with Col_f4
int Hornpipe_for_Lisa_tune[] = {Col_e4, Col_g4, Col_f4, Col_b4, Col_c4, Col_a4m, Col_b4m,
Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_f4, Col_c4, Col_a4m, Col_b4m};
int Hornpipe_for_Lisa_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(Hornpipe_for_Lisa_tune) / 2; // Tune length i.o.t. loop
Hornpipe for Lisa added NO_OF_REPETETIONS No - 5
*/
int StartingBuzzer_tune[] = {Col_c5, Col_b4m, Col_a4m, Col_e4, Col_c4};
int StartingBuzzer_beats[] = {20, 20, 20, 20, 20};
int StartingBuzzer_MAXIMUM_COUNT = sizeof(StartingBuzzer_tune) / 2; // Tune length i.o.t.
loop
// Starting Buzzer changed No of repetitions to 01
int NO_OF_REPETITIONS = 1;
int MAXIMUM_COUNT = 0;
// tempo
long tempo = 35000;  // Ver3 for Greensleeves original tempo lengthened from 10000 to 30000
                     // Hornpipe for Lisa tempo lenghtened from 30000 to 35000
// length of pause
int pause  = 10000; // original pause increased from 1000 to 5000
                    // Hornpipe for Lisa pause increased from 5000 to 10000
// add to loop to pause for 1 sec
int rest_count = 200;
                    // Hornpipe for Lisa rest_count increased from 100 to 200
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;
```

```
// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Colmans Scale - Try to play the entire scale
// Hornpipe for Lisa added next line
int reps = 1;
int music = 1;
void loop(){
  if (reps < NO_OF_REPETITIONS){
      if (music == 1){
        MAXIMUM_COUNT = StartingBuzzer_MAXIMUM_COUNT;
      }
      if (music == 2){
        MAXIMUM_COUNT = colmans_MAXIMUM_COUNT;
      }
      if (music == 3){
        MAXIMUM_COUNT = Colmans_hornpipe_MAXIMUM_COUNT;
      }
      for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
        if (music == 2){
          note = colmans_tune[timer];
          beat = colmans_beats[timer];
        }
        if (music == 1){
          note = StartingBuzzer_tune[timer];
          beat = StartingBuzzer_beats[timer];
        }
        if (music == 3){
          note = Colmans_hornpipe_tune[timer];
          beat = Colmans_hornpipe_beats[timer];
        }
```

```
        duration = beat * tempo;
        play_note();
        delayMicroseconds(pause);
      }
      reps ++;
  }
   else{
      digitalWrite(buzzer_out, LOW);
/* Debugging removed for Hornpipe for Lisa
    if (DEBUGGING){
      Serial.println(timer);
      Serial.println("times round loop:");
      Serial.println(note);
      Serial.println("-note");
      Serial.println(beat);
      Serial.println("-beat");
      Serial.println(duration);
      Serial.println("-duration");
    }
*/
  }
}
```

Annex B – Appendix 14 - Scetch_nov23a_MusicVer02

This version was submitted to the team.
Note for appendices I removed all ancillary spaces to save space, however here I am showing how I submitted it so I am leaving the spaces in.

```
/* Colman O'Keeffe 114712191 26 Nov 2016 - Final version to be copied into Main Program - Comments deleted to shorten code

# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4m 3214 // 311.13 Hz
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4m 2407 // 415.30 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4m 2025 // 493.88 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define Col_a5 1136 // 880.00 Hz
# define Col_b5 1012 // 987.77 Hz

# define rest 0

int buzzer_out = 12;

void setup(){
  pinMode(buzzer_out, OUTPUT);
}

int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4, Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {30, 16, 20, 10, 40, 20, 60, 20, 40, 10, 40, 20, 30, 10, 60, 20};
int colmans_MAXIMUM_COUNT = sizeof(colmans_tune) / 2;

int Colmans_hornpipe_tune[] = {Col_e4, Col_c5, Col_f4, Col_d4, Col_c4, Col_a4m, Col_b4m, Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_d4, Col_c4, Col_a4m, Col_b4m};
int Colmans_hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int Colmans_hornpipe_MAXIMUM_COUNT = sizeof(Colmans_hornpipe_tune) / 2;

int StartingBuzzer_tune[] = {Col_c5, Col_b4m, Col_a4m, Col_e4, Col_c4};
int StartingBuzzer_beats[] = {20, 20, 20, 20, 20};
int StartingBuzzer_MAXIMUM_COUNT = sizeof(StartingBuzzer_tune) / 2; // Tune length i.o.t. loop

int NO_OF_REPETITIONS = 1;
int MAXIMUM_COUNT = 0;
```

```
long tempo = 35000;
int pause  = 10000;
int rest_count = 200;

int note = 0;
int beat = 0;
long duration = 0;

// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}

int reps = 1;
int music = 1;

void loop(){
  if (reps < NO_OF_REPETITIONS){
      if (music == 1){
        MAXIMUM_COUNT = StartingBuzzer_MAXIMUM_COUNT;
      }
      if (music == 2){
        MAXIMUM_COUNT = colmans_MAXIMUM_COUNT;
      }
      if (music == 3){
        MAXIMUM_COUNT = Colmans_hornpipe_MAXIMUM_COUNT;
      }
      for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
        if (music == 2){
          note = colmans_tune[timer];
          beat = colmans_beats[timer];
        }
```

```
        if (music == 1){
          note = StartingBuzzer_tune[timer];
          beat = StartingBuzzer_beats[timer];
        }
        if (music == 3){
          note = Colmans_hornpipe_tune[timer];
          beat = Colmans_hornpipe_beats[timer];
        }
        duration = beat * tempo;
        play_note();
        delayMicroseconds(pause);
      }
      reps ++;
    }
     else{
       digitalWrite(buzzer_out, LOW);
    }
}
```

Annex B – Appendix 15 - Scetch_nov23a_MusicVer03

Yes, they didn't like it either, because they couldn't get it to work in their loop.
I had to add a start variable.
This was the FINAL version submitted to the team.

```
/* Colman O'Keeffe 114712191 23 Nov 2016 - Completed 26 Nov 2016 at 1815
* note frequencies from http://www.phy.mtu.edu/~suits/notefreqs.html 1540 23 Nov 2016
Ver 01 period = 1/f in milli secs all values calculated manually on calculator
Ver 02 all periods altered randomly to ensure a know tune is NOT used
the differance in the change in tones is not noticable to me as a layman.
Ver 3 returned note values to original values added A4 minor, B4 minor, E4 Minor & B5
Hornpipe Ver01 added A5
*/

# define Col_c4 3822 // 261.63 Hz period = 1/f in milli secs
# define Col_d4 3405 // 293.66 Hz all values calculated manually on calculator
# define Col_e4m 3214 // 311.13 Hz
# define Col_e4 3037 // 329.63 Hz
# define Col_f4 2863 // 349.23 Hz
# define Col_g4 2551 // 392.00 Hz
# define Col_a4m 2407 // 415.30 Hz
# define Col_a4 2272 // 440.00 Hz
# define Col_b4m 2025 // 493.88 Hz
# define Col_b4 2025 // 493.88 Hz
# define Col_c5 1911 // 523.25 Hz
# define Col_a5 1136 // 880.00 Hz
# define Col_b5 1012 // 987.77 Hz

# define rest 0

// buzzer
int buzzer_out = 12;

// debugging serial out? 1 yes 0 no
int DEBUGGING = 0;

void setup(){
  pinMode(buzzer_out, OUTPUT);
/* debugging removed for Hornpipe for Lisa
  if (DEBUGGING){
    Serial.begin(9600); // serial out for debug code
  }
*/
}
/* Colmans tune test scale
int colmans_tune[] = {Col_c4, Col_d4, Col_e4, Col_f4, Col_g4, Col_a4, Col_b4, Col_c5};
int colmans_beats[] = {16, 16, 16, 16, 16, 16, 16, 16};
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
```

```
*/
// Ver 1 change scale notes in random order altered length of beats in random order
// Ver 2a doubled length of notes in tune added random beats

/* First attempt at music
int colmans_tune[] = {Col_c4, Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4, Col_c4,
Col_g4, Col_d4, Col_f4, Col_a4, Col_c5, Col_b4, Col_e4};
int colmans_beats[] = {30, 16, 20, 10, 40, 20, 60, 20, 40, 10, 40, 20, 30, 10, 60, 20};
int colmans_MAXIMUM_COUNT = sizeof(colmans_tune) / 2;
int MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop
*/

// Ver 3 attempt at Greensleeves
// chords from https://www.acousticmusicarchive.com/greensleeves-chords-lyrics
int colmans_tune[] = {Col_g4, Col_d4, Col_b4m, Col_a4m, Col_b4, Col_g4, Col_d4, Col_b4m,
Col_c4, Col_b5, Col_e4m};
int colmans_beats[] = {60, 20, 20, 60, 40, 60, 20, 40, 40, 10, 30};
int colmans_MAXIMUM_COUNT = sizeof(colmans_tune) / 2; // Tune length i.o.t. loop

/* HornpipeVer01 http://www.guitarnick.com/sailors-hornpipe-irish-flatpicking-guitar-score-
tab.html

int hornpipe_tune[] = {Col_g4, Col_a5, Col_d4, Col_g4, Col_c4, Col_d4, Col_g4, Col_d4, Col_g4,
Col_g4, Col_c4, Col_a5, Col_d4,  Col_g4, Col_c4, Col_d4, Col_g4};
int hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(hornpipe_tune) / 2; // Tune length i.o.t. loop
*/

int Colmans_hornpipe_tune[] = {Col_e4, Col_c5, Col_f4, Col_d4, Col_c4, Col_a4m, Col_b4m,
Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_d4, Col_c4, Col_a4m, Col_b4m};
int Colmans_hornpipe_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int Colmans_hornpipe_MAXIMUM_COUNT = sizeof(Colmans_hornpipe_tune) / 2;

/* Hornpipe for Lisa - replaced Chord02 Col_c5 with Col_g4 - replaced Chord04 Col_d4 with
Col_b4 - replaced Chord14 with Col_g4 with Col_f4
int Hornpipe_for_Lisa_tune[] = {Col_e4, Col_g4, Col_f4, Col_b4, Col_c4, Col_a4m, Col_b4m,
Col_a4m, Col_b4m, Col_e4, Col_c4, Col_e4, Col_c4, Col_f4, Col_c4, Col_a4m, Col_b4m};
int Hornpipe_for_Lisa_beats[] = {40, 20, 20, 20, 20, 10, 8, 10, 8, 20, 20, 20, 20, 10, 10, 10, 10};
int MAXIMUM_COUNT = sizeof(Hornpipe_for_Lisa_tune) / 2; // Tune length i.o.t. loop
Hornpipe for Lisa added NO_OF_REPETETIONS No - 5
*/
int StartingBuzzer_tune[] = {Col_c5, Col_b4m, Col_a4m, Col_e4, Col_c4};
int StartingBuzzer_beats[] = {20, 20, 20, 20, 20};
int StartingBuzzer_MAXIMUM_COUNT = sizeof(StartingBuzzer_tune) / 2; // Tune length i.o.t.
loop
// Starting Buzzer changed No of repetitions to 01
int NO_OF_REPETITIONS = 1;
int MAXIMUM_COUNT = 0;
```

```
// tempo
long tempo = 35000;  // Ver3 for Greensleeves original tempo lengthened from 10000 to 30000
                     // Hornpipe for Lisa tempo lenghtened from 30000 to 35000
// length of pause
int pause  = 10000; // original pause increased from 1000 to 5000
                    // Hornpipe for Lisa pause increased from 5000 to 10000
// add to loop to pause for 1 sec

int rest_count = 200;
                    // Hornpipe for Lisa rest_count increased from 100 to 200
// setup variables for main function
int note = 0;
int beat = 0;
long duration = 0;

// try to play a note
void play_note(){
  long time_so_far = 0;
  if (note > 0){
    while(time_so_far < duration){
      // buzzer on
      digitalWrite(buzzer_out, HIGH);
      delayMicroseconds(note / 2);
      // buzzer off
      digitalWrite(buzzer_out, LOW);
      delayMicroseconds(note / 2);
      // how much time has gone by
      time_so_far += (note);
    }
  }
  else{
    for (int restbeat = 0; restbeat < rest_count; restbeat++){
      delayMicroseconds(duration);
    }
  }
}
// Colmans Scale - Try to play the entire scale
// Hornpipe for Lisa added next line
// MusicVer01 reps only 1 for each tune.
int reps = 0;
int music = 3;
int start = 0;
void loop(){
  if (reps < NO_OF_REPETITIONS){
      if ((music == 1) && (start == 0)){
        MAXIMUM_COUNT = StartingBuzzer_MAXIMUM_COUNT;
        start = 1;
      }
      if ((music == 2) && (start == 0)){
```

```
      MAXIMUM_COUNT = colmans_MAXIMUM_COUNT;
      start = 1;
     }
     if ((music == 3) && (start == 0)){
      MAXIMUM_COUNT = Colmans_hornpipe_MAXIMUM_COUNT;
      start = 1;
     }
     for (int timer = 0; timer < MAXIMUM_COUNT; timer++){
      if (music == 1){
       note = StartingBuzzer_tune[timer];
       beat = StartingBuzzer_beats[timer];
      }
      if (music == 2){
       note = colmans_tune[timer];
       beat = colmans_beats[timer];
      }
      if (music == 3){
       note = Colmans_hornpipe_tune[timer];
       beat = Colmans_hornpipe_beats[timer];
      }
      duration = beat * tempo;
      play_note();
      delayMicroseconds(pause);
     }
     reps ++;
  }
   else{
     digitalWrite(buzzer_out, LOW);
/* Debugging removed for Hornpipe for Lisa
    if (DEBUGGING){
     Serial.println(timer);
     Serial.println("times round loop:");
     Serial.println(note);
     Serial.println("-note");
     Serial.println(beat);
     Serial.println("-beat");
     Serial.println(duration);
     Serial.println("-duration");
    }
*/
  }
}
```

Annex C – Project Plan Version 01 – It was not necessary to write another as this worked.

Project Plan

written by Herakliusz Lipiec, John O'Dowd and Colman O'Keeffe

Tic Tac Toe for the Arduino

We used an agile system.

We began with group discussions as to the building of the hardware circuit and what methods we would use to write the code to determine whether we could efficiently determine a win or a draw.

We discussed whether using two 1 : 16 demultiplexers would be a more efficient method of connecting to each of the 9 Patterns of LEDs and had a good discussion about whether a three person game was possible with a 4x4 grid. This would require a more efficient method of determining a winner.

Colman decided that he would build the circuit as described by the lecturer, this was later ammended by Herakliusz in order to better suit his code.

Herakliusz wanted to write the i/o code and later decided to implement a GUI.

John wanted to do the theory on how to determine the winner.

Colman wanted to write the music, when I saw that it was a requirement for the project, i.e. the victory tune.

----------End of Doc----------

Note:
Reverse Alphabetical because it is unfair to always have the person with the lowest charachter first.
I changed one letter because my version of Apache Open Office does not have a spell checker.
I typed the plan. We all agreed on it.

Annex D – Music

Greensleeves - Traditional
chords, lyrics and origins from https://www.acousticmusicarchive.com/greensleeves-chords-lyrics
Note: when I copied and pasted from this source, it put a lovely script into my document, which I then deleted.

Original notation at

http://www.musicksmonument.com/Hans_Holbein/HOLBEIN_-_GREENSLEEVES.html

Greensleeves - Chords, Lyrics and Origins

Origins

Mentioned in more than one Shakespeare play, Greensleeves is a traditional English folk song that probably dates from the late sixteenth century. A quick glance at the lyrics, in which a man vainly and somewhatobsessively courts an unobtainable woman, explains why people have assumed that it was written by Henry VIII, about his ultimately successful, but nevertheless long-drawn out and rather tortuous courtship of Ann Boleyn. However, there is no evidence for this. What's for sure is that courtly love in all its unobtainability (if that's a word) was a major theme of much Renaissance poetry. And that's probably the tradition that this set of lyrics draws from.

Chords

Capo at 2nd Fret
Verse
Em G D Bm
Alas, my love, you do me wrong,
C B
To cast me off discourteously.
Em G D Bm
For I have loved you so long,
C B7 Em
Delighting in your company.

Chorus
G D Bm
Greensleeves was all my joy
Am B
Greensleeves was my delight,
G D Bm

Greensleeves was my heart of gold,
C B7 Em
And who but my lady greensleeves?

Lyrics

Alas, my love, you do me wrong,
To cast me off discourteously.
For I have loved you so long,
Delighting in your company.

Greensleeves was all my joy
Greensleeves was my delight,
Greensleeves was my heart of gold,
And who but my lady greensleeves?

Your vows you've broken, like my heart,
Oh, why did you so enrapture me?
Now I remain in a world apart,
But my heart remains in captivity.

Greensleeves was all my joy
Greensleeves was my delight,
Greensleeves was my heart of gold,
And who but my lady greensleeves?

I have been ready at your hand,
To grant whatever you would crave;
I have both wagered life and land,
Your love and good-will for to have.

Greensleeves was all my joy
Greensleeves was my delight,
Greensleeves was my heart of gold,
And who but my lady greensleeves?

If you intend thus to disdain,
It does the more enrapture me,
And even so, I still remain
A lover in captivity.

Greensleeves was all my joy
Greensleeves was my delight,
Greensleeves was my heart of gold,
And who but my lady greensleeves?

My men were clothed all in green,
And they did ever wait on thee;
All this was gallant to be seen,
And yet thou wouldst not love me.

Greensleeves was all my joy
Greensleeves was my delight,
Greensleeves was my heart of gold,
And who but my lady greensleeves?

Thou couldst desire no earthly thing,
but still thou hadst it readily.
Thy music still to play and sing;
And yet thou wouldst not love me.

Greensleeves was all my joy
Greensleeves was my delight,
Greensleeves was my heart of gold,
And who but my lady greensleeves?

Well, I will pray to God on high,
that thou my constancy mayst see,
And that yet once before I die,
Thou wilt vouchsafe to love me.

Greensleeves was all my joy
Greensleeves was my delight,
Greensleeves was my heart of gold,
And who but my lady greensleeves?

Ah, Greensleeves, now farewell, adieu,
To God I pray to prosper thee,
For I am still thy lover true,
Come once again and love me.

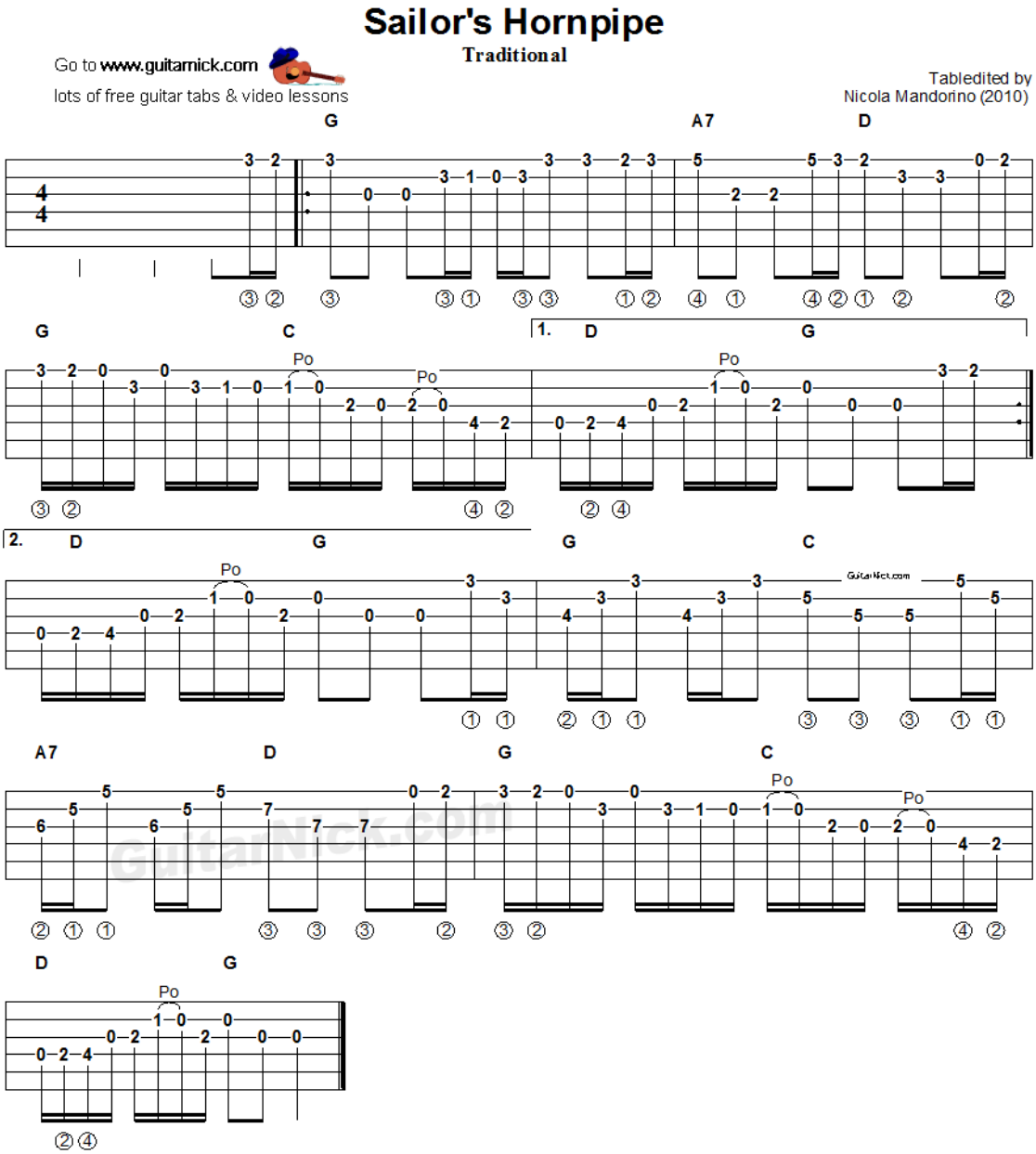
Greensleeves was all my joy
Greensleeves was my delight,

Greensleeves was my heart of gold,
And who but my lady greensleeves?

The hornpipe I wrote was adapted from the following site:

http://www.guitarnick.com/sailors-hornpipe-irish-flatpicking-guitar-score-tab.html

Based on:

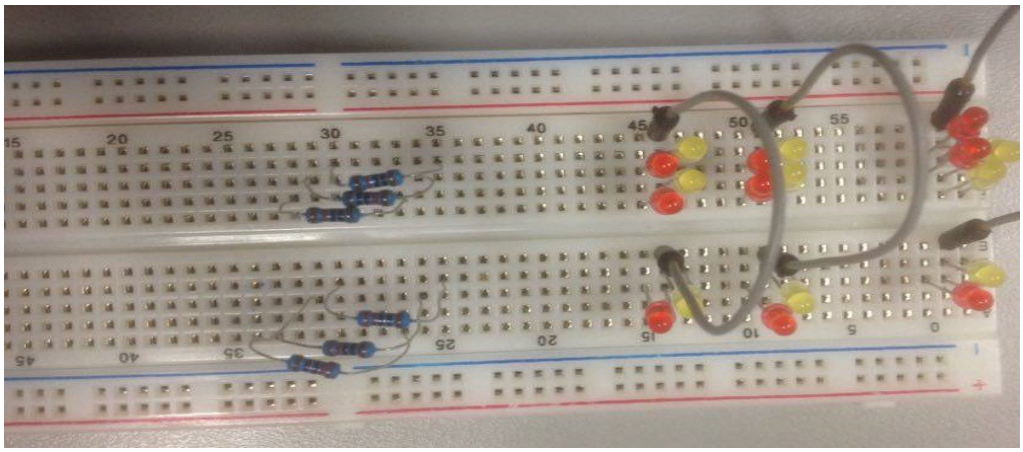


Written by Colman O'Keeffe 114712191
Assignment 02 Tic-Tac-Toe
University College Cork Module CS3514 C Programming for MicroControllers

Annex D – The circuit build.

Photos of the Circuit Build:

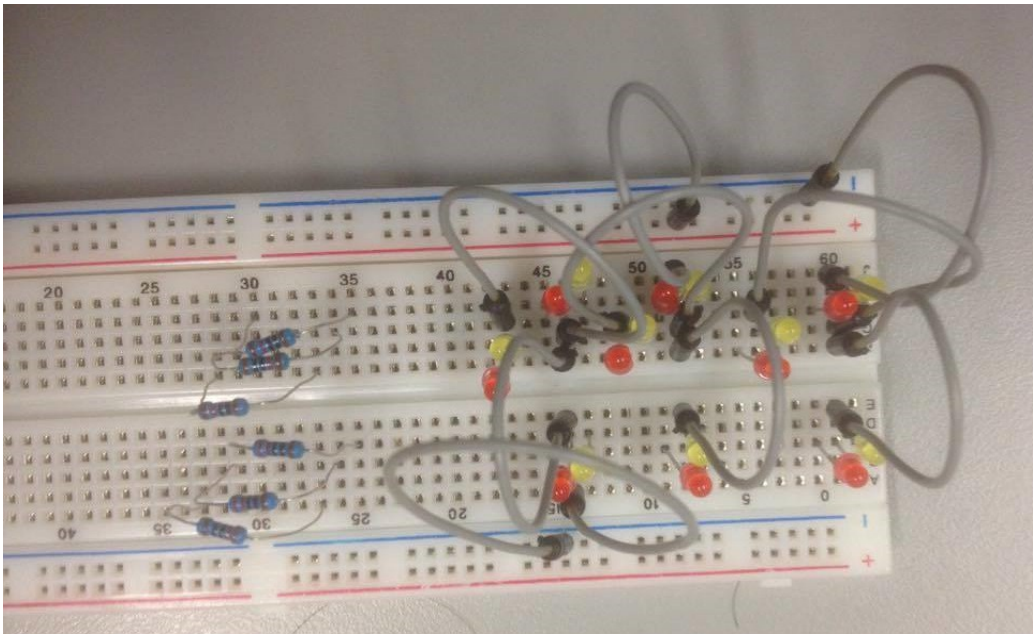
Photo 01



Initial Setup

Note: LEDS were placed into the incorrect pins see line 45 for example. 2 reds into the same line.

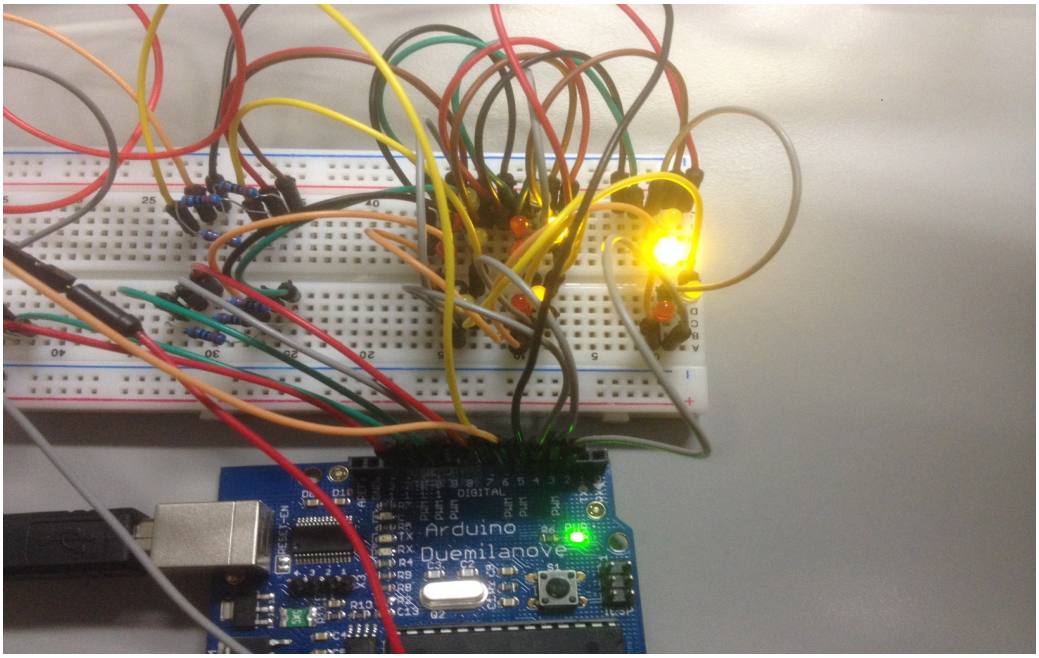
Photo 02



First mistake rectified

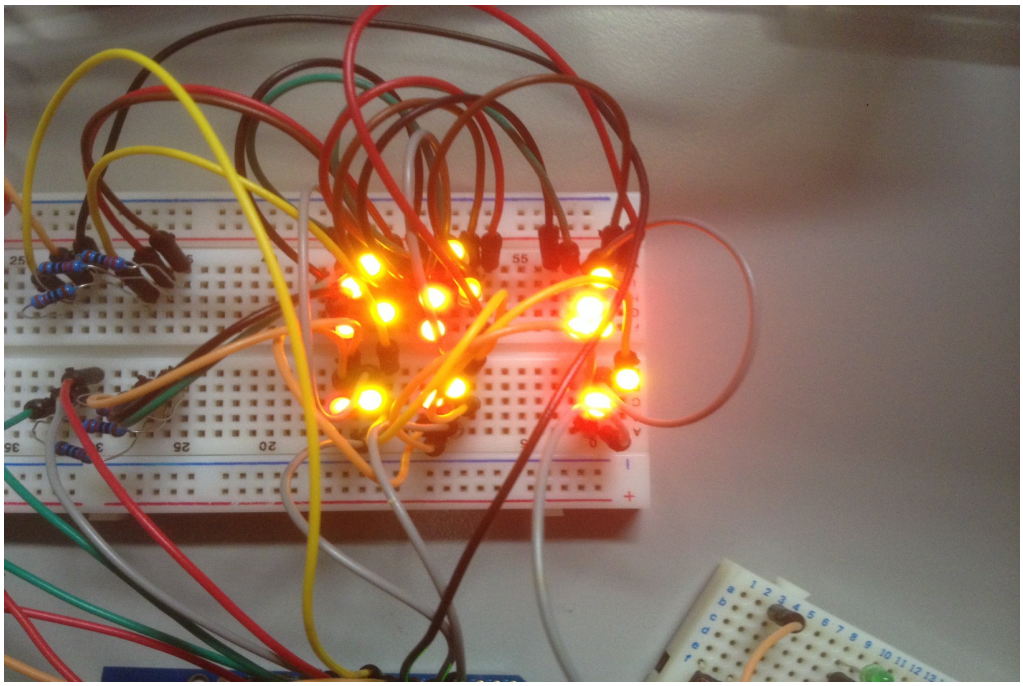
Note: LEDS now in correct lines, however spacing makes it hard to correctly interpret the board, I didnt want to bend the pins on the LEDs. (Herakliusz wanted them changed, he was correct.)

Photo 03



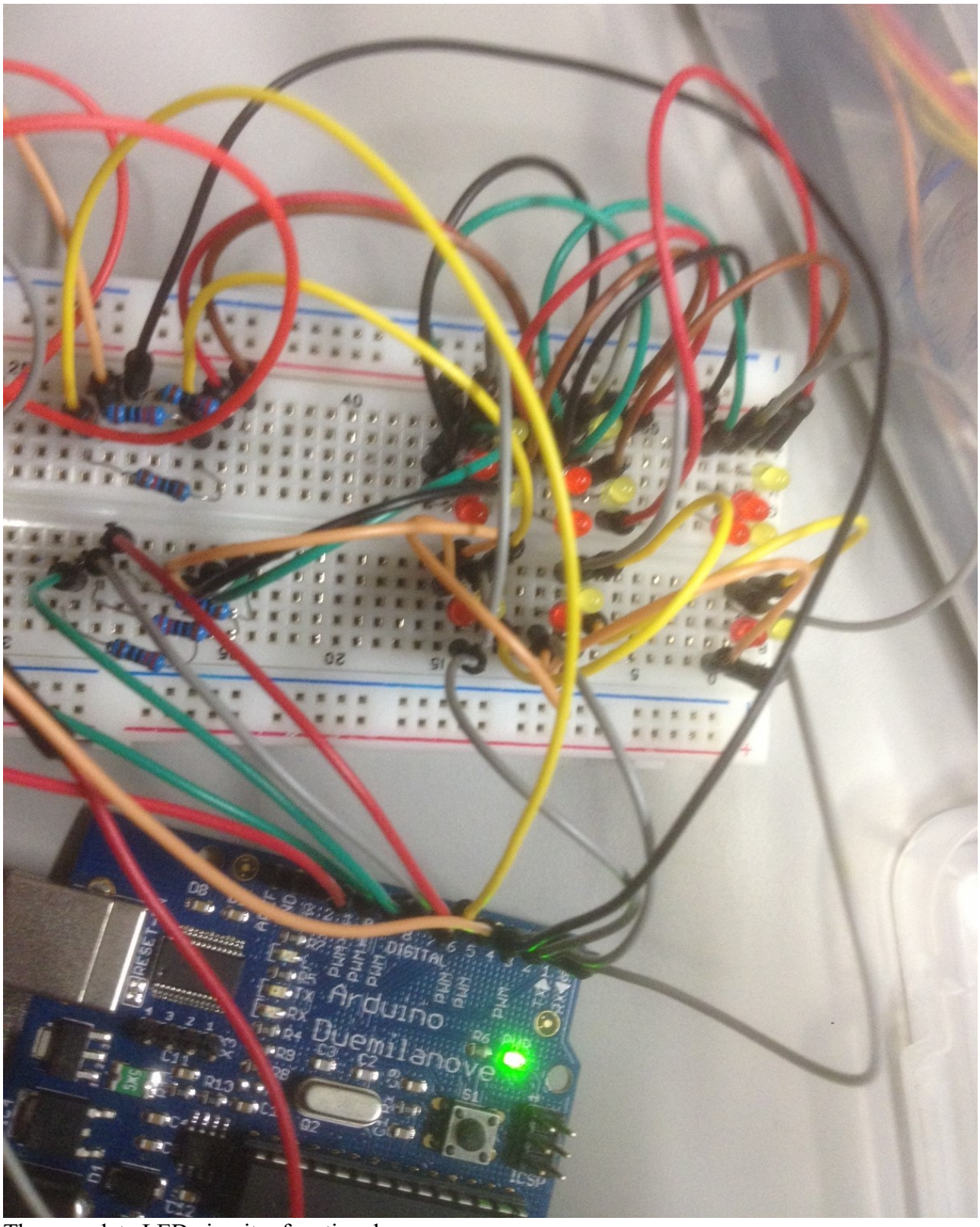
Testing the LED circuit
Note: Test code by Herakliusz

Photo 04



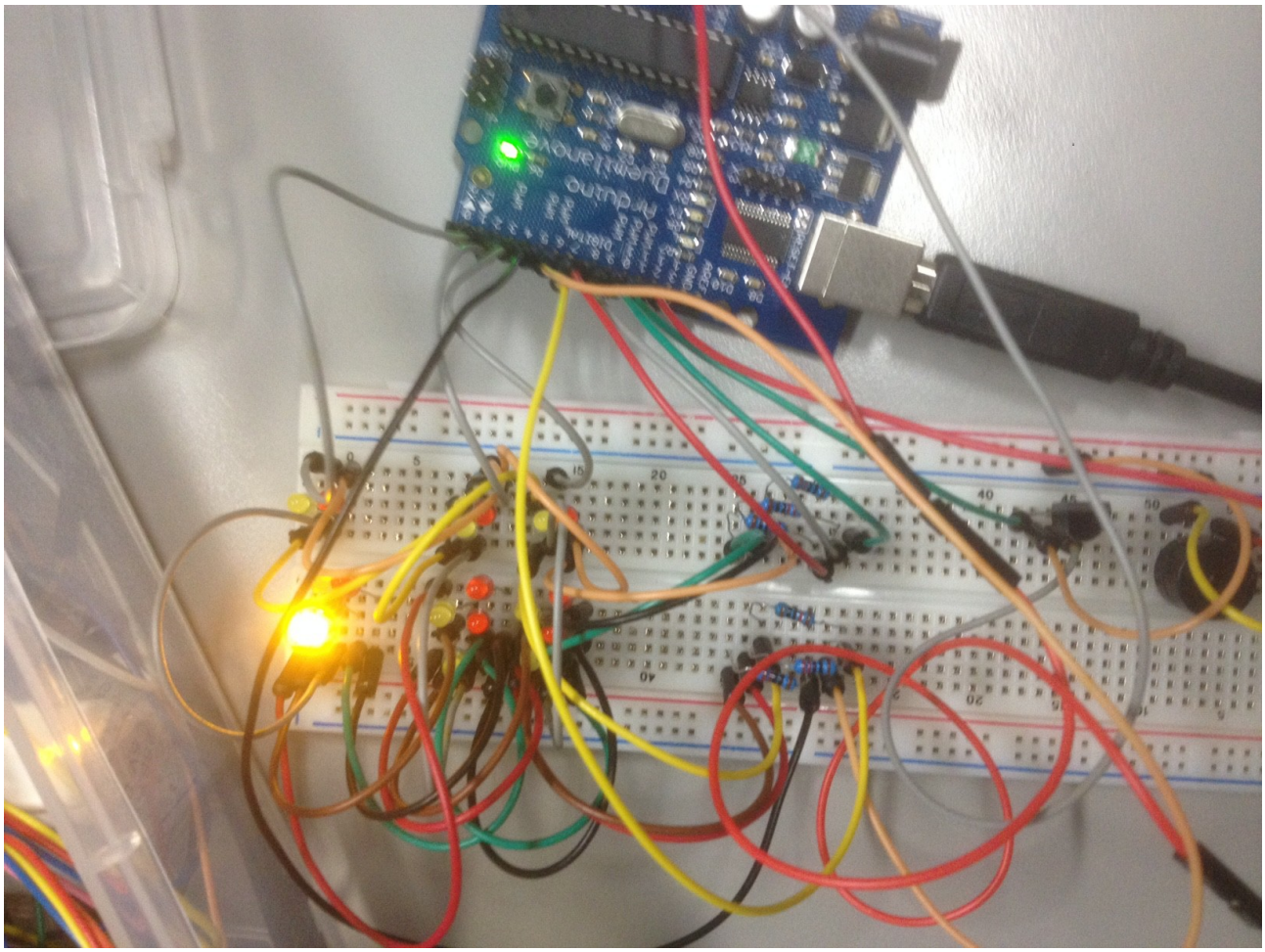
Testing the LED circuit – All LEDs functioning.
Note: The iPhone camera (Apple™ iPhone4) does not properly differentiate between yellow and red LEDs

Photo 05



The complete LED circuit – functional.

Photo 06

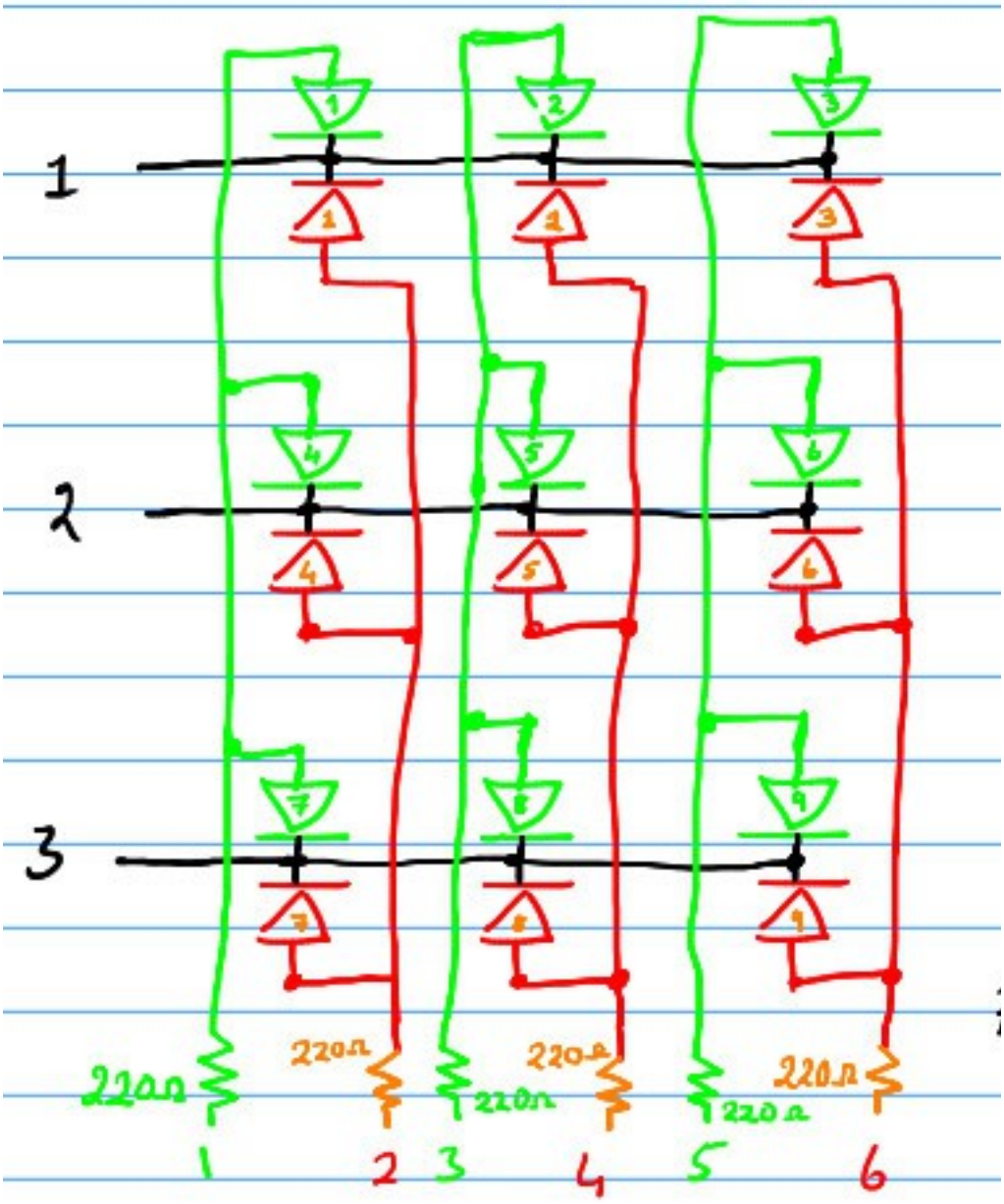


The complete function circuit including IR reciever and Buzzer. Taken during a test.

Note: There were no real problems building the circuit. One 10,000 Ohm resistor was inserted by accident, I tested all the resistors before I inserted them, however for some reason there was one in there. I should have used the carpenter's adage, “measure twice, cut once”.

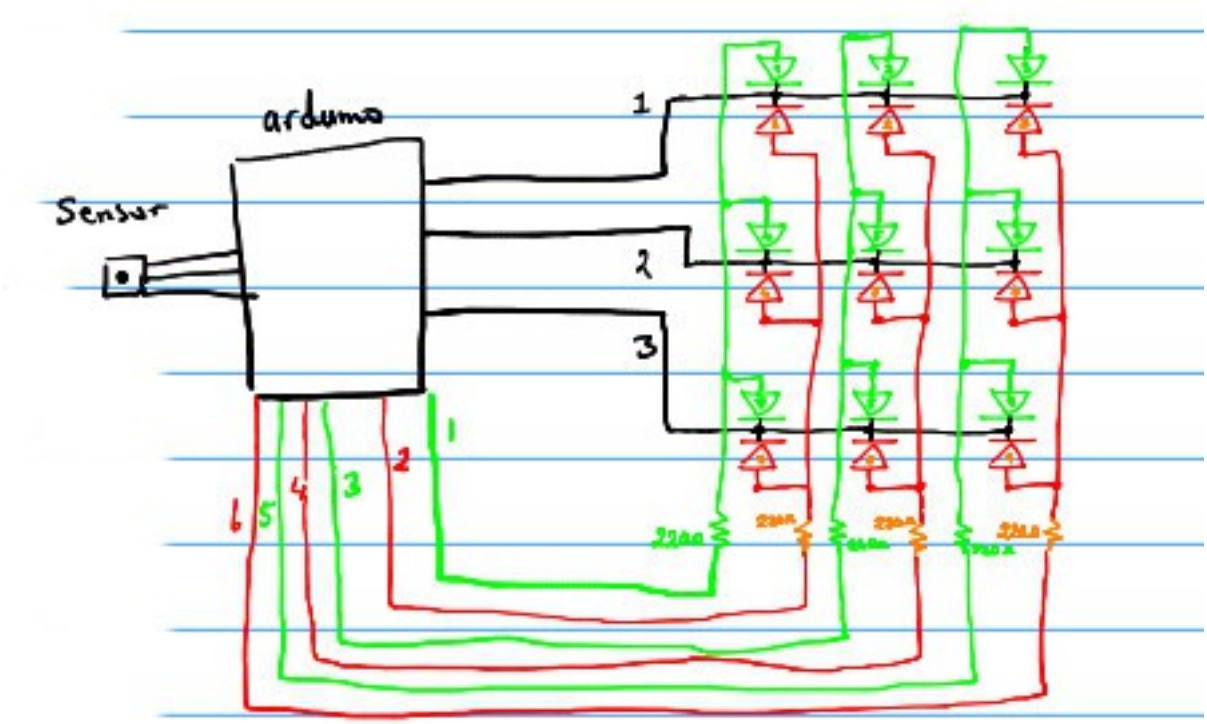
Annex E - Diagrams describing the functioning of the circuit as provided by Lecturer.

Diagram 01



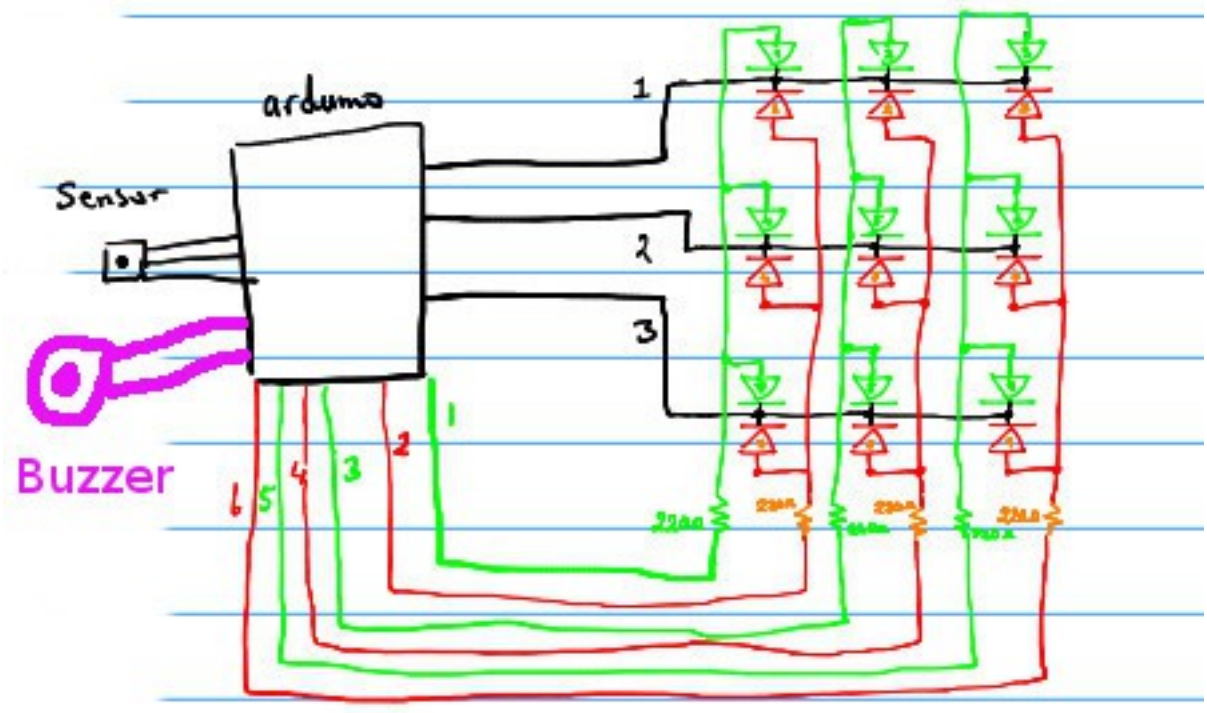
Electronics diagram

Diagram 02



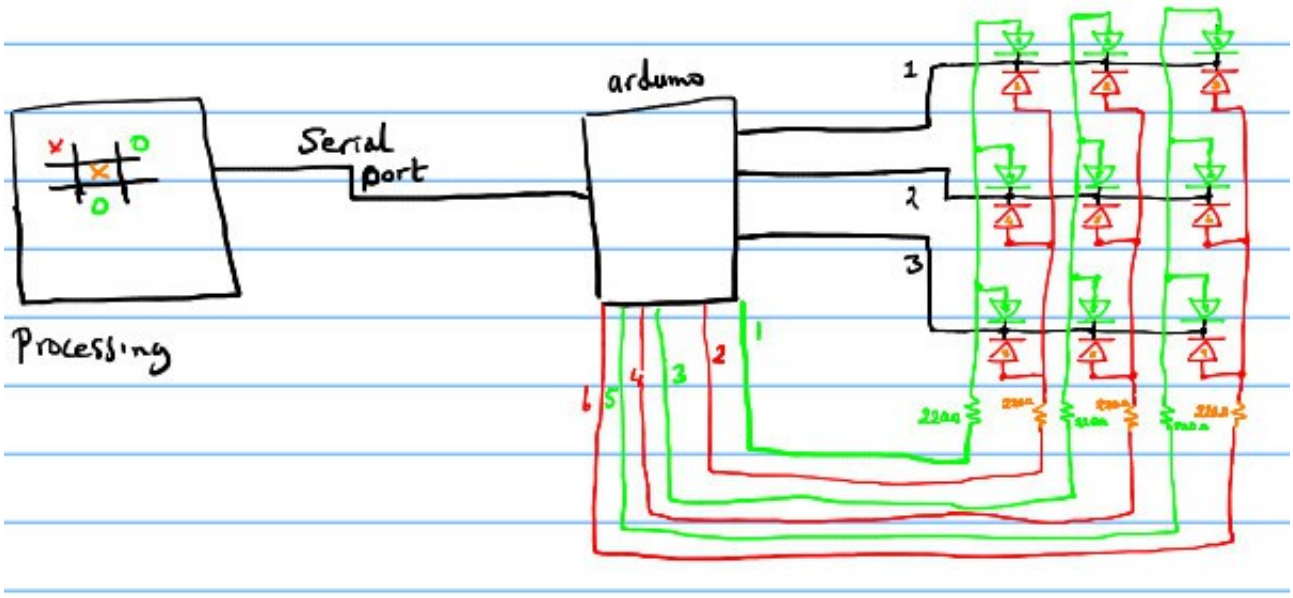
Connecting the Arduino to the Circuit
Note the buzzer was not included.

Diagram 03



Lecturers Diagram 02 altered by myself using Gimp 2 to show Buzzer in purple.

Diagram 04



Connecting the arduino to the GUI on a separate independant processor.

Note: With a little effort, it should be possible to link it over the internet. It might take a Herakliusz a little time though.

Annex F – The evolution of the Tic-tac-toe Manual – Three versions.
Scan 01 - Scanner Input (Epson Perfection V500 Photo). the Tic-tac-toe Manual Version 01

Version 02

Tic – Tac – Toe for the Auduino Manual
by Colman, John and Herakliusz in C for Microcontrollers CS3514

Rules for TicTacToe,(noughts and crosses): A game for Two players.

The beginning player Player One is traditionally selected by a coin toss or by mutual arrangement.

Each player selects positions alternatively on a # shaped board. The same postion may NOT be selected by both players

A win is achieved by making a line of three identical symbols horizontally or diagonally.

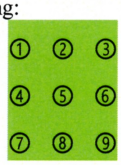
The game may finish with a draw where no winning combination is reached and all positions are selected.

In subsequent games the players traditionally alternate beginning as Player One.

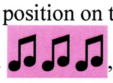
In our game the symbols are represented by yellow and red LEDs or Buttons on the GUI.

Start:

Press the Θ Play/Pause Button – A tune will be played  .
To select Player Ones colour press the MODE Button –
The colour to play first will show 9 LEDs.
Using the *remote*:
On the remote select your position on the board by pressing:



Using the *mouse*:
Click on the button representing that position on the board.

If a player wins a tune will be played  , the winners LEDs will all light up.
In order to see the resulting board for the last game, press the ↺ Button.

Please Note:

If an invalid selection is made a nasty sound will be made  . The game will await a valid selection.

The game may be saved at any time by pressing the ⓪ Button.

The sound may be muted at any time by pressing the & SOUND Button.
The LEDs may be function checked by pressing MODE two times after the start in order to ensure all 18 LEDS are functional.

Page No 1 of 1
Manual of TicTacToe Ver 02 16:47:10 Tuesday 29 November 2016
Authors: Heraclius Lipiec, John O'Dowd, Colman O'Keeffe

On this version Herakliusz did not like the Play/Pause symbol or the return button. Also some buttons were incorrect. We discussed it on the screen and I took away the file to upgrade it. Note writing on it was an error.

Tic – Tac – Toe for the Auduino Manual
by Colman, John and Herakliusz in C for Microcontrollers CS3514

Rules for TicTacToe,(noughts and crosses): A game for Two players.

The beginning player Player One is traditionally selected by a coin toss or by mutual arrangement.

Each player selects positions alternatively on a # shaped board. The same postion may NOT be selected by both players

A win is achieved by making a line of three identical symbols horizontally or diagonally.

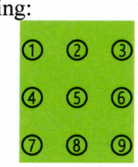
The game may finish with a draw where no winning combination is reached and all positions are selected.

In subsequent games the players traditionally alternate beginning as Player One.

In our game the symbols are represented by yellow and red LEDs or Buttons on the GUI.

Start:

Press the Play/Pause Button – A tune will be played .
To select Player Ones colour press the MODE Button –
The colour to play first will show 9 LEDs.
Using the *remote*:
On the remote select your position on the board by pressing:



Using the *mouse*:
Click on the button representing that position on the board.

If a player wins a tune will be played , the winners LEDs will all light up.
In order to see the resulting board for the last game, press the Button.

Please Note:

If an invalid selection is made a nasty sound will be made . The game will await a valid selection.

The game may be saved at any time by pressing the EQ Button.

The sound may be muted at any time by pressing the SOUND Button.
The LEDs may be function checked by pressing MODE two times after the start in order to ensure all 18 LEDS are functional.

Page No 1 of 1
Manual of TicTacToe Ver 02 16:47:10 Tuesday 29 November 2016
Authors: Heraclius Lipiec, John O'Dowd, Colman O'Keeffe

I changed the buttons which were incorrect and I drew buttons which I could not find in the standard fonts with Gimp 2. from https://www.gimp.org/, beware the other download sites purporting to be from Gimp.

Scan 03 – the Tic-tac-toe Manual Version 03

Tic – Tac – Toe for the Auduino Manual
by Colman, John and Herakliusz in C for Microcontrollers CS3514

Rules for TicTacToe,(noughts and crosses): A game for Two players.

The beginning player Player One is traditionally selected by a coin toss or by mutual arrangement.

Each player selects positions alternatively on a # shaped board. The same postion may NOT be selected by both players

A win is achieved by making a line of three identical symbols horizontally or diagonally.

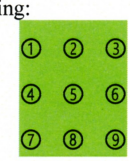
The game may finish with a draw where no winning combination is reached and all positions are selected.

In subsequent games the players traditionally alternate beginning as Player One.

In our game the symbols are represented by yellow and red LEDs or Buttons on the GUI.

Start:

Press the Play/Pause Button – A tune will be played .
To select Player Ones colour press the MODE Button –
The colour to play first will show 9 LEDs.
Using the *remote*:
On the remote select your position on the board by pressing:



Using the *mouse*:
Click on the button representing that position on the board.

If a player wins colmans tune will be played , the winners LEDs will all light up.

If there is a draw Greensleeves will be played.
In order to see the resulting board for the last game, press the Button.

Please Note:

If an invalid selection is made a nasty sound will be made . The game will await a valid selection.

The game may be saved at any time by pressing the EQ Button.

The sound may be muted at any time by pressing the SOUND Button.
The LEDs may be function checked by pressing MODE two times after the start in order to ensure all 18 LEDS are functional.

Page No 1 of 1
Manual of TicTacToe Ver 03 06:48:52 Thursday 15 December 2016
Authors: Heraclius Lipiec, John O'Dowd, Colman O'Keeffe

This is the final version uploaded to Git Hub.
I changed it after noticed we had not differeciated our victory tune from our draw tune (or mentioned our draw tune).

Written by Colman O'Keeffe 114712191
Assignment 02 Tic-Tac-Toe
University College Cork Module CS3514 C Programming for MicroControllers

1100hrs Lab G21 21Nov 2016
Testing the IR reciever
FFFFFFFF is sent after the IR reciever sends the code for the button tested.
The remote continues to send this until the button is released.

The buzzer sounded as the program dictated on pin 12.

To make a tune use an array of time on, time off for buzzer.

We wrote test code to make the leds light up to test the circuit was correct.
facebook code for yellow

for red change 567 to 8,9,10
We wrote code to test the red leds

We built the circuit on the emulator circuit.io see attached diagram

We tested the circuit on the emulator for yellow leds

we tested the circuit on the emulator for red leds

Note: We should have been doing this constantly (in an experimental notebook), however we only did it for the one lab.