DREAMINO CYBER LEARNING SYSTEM

David Ray Electronics and More LLC

Booklet v0.02.01 – 20191019

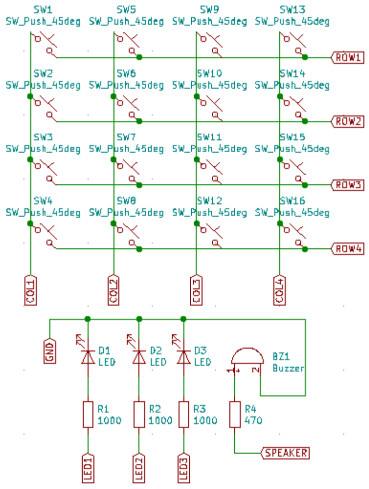
David Ray and Angel Andino

<u>Book Four</u>

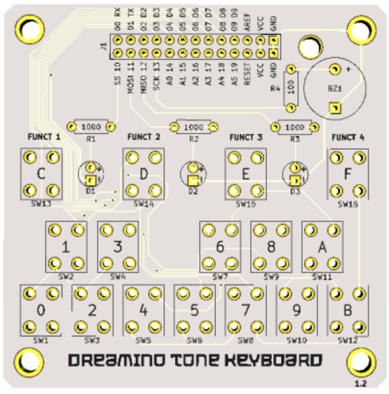
Section 1: The Tone Keyboard

The Tone Keyboard kit is an addon that goes on to your starter kit. It has a speaker, three LEDs, and 16 buttons. The buttons are laid out like a normal musical keyboard.

1.01: Button Matrix – The keyboard is laid out in a button matrix. Normally we would need to use a single pin for each button, meaning we would need to use 16 digital pins to control the buttons, but by using a matrix we are able to use all 16 buttons with only 8 pins on our micro-controller. To do this, we need to place the buttons in a grid pattern, segmenting the buttons into individual rows and columns. With them broken down in that way, we can just check each row for a button being pressed, then move to the next row, etc.



Normally when we use a button we check the button state with a 'digitalRead()' function. We see if it is LOW or HIGH and then have the boards act accordingly, but in a matrix we activate one row at a time then check each column of that row. I know this sounds a little complicated, but luckily someone wrote a library called 'Keypad' to do all the work for you.



Rows	Pin	Columns	Pin
Row 1	8	Column 1	7
Row 2	9	Column 2	4
Row 3	3	Column 3	2
Row 4	5	Column 4	6

The board has some other functions. The three LEDs are individually addressable and there a speaker to create sounds. We will go over that a bit more later, when we start writing for the board.

Function	**Pin**
LED 1	10
LED 2	11
LED 3	12
Speaker	13

Section 2: Introduction to Libraries

A library is basically a small file or group of files that make it easier to share functionality among different programs. For example, earlier we used the two libraries to add screen functionality. The 'Adafruit_GPX' and 'Adafruit_PCD8544' were libraries that were written by a company named Adafruit that allows use to use the screen on the gamer board. We could have used the screen without a library, but it would have been a lot more difficult and would take a lot more time. Every library has a header file that is ends with a '.h' extension. In the session we will go over how to use header files.

2.01: Header Files – In this session we will be using the Tone Keyboard add-on kit. In earlier sessions we went over variables, like 'int led = 13;'. These were statements like this that were in our main program, but if we were going to use the same variable sets in many different programs, we could just copy the entire variable set into another program easily. For the tone keyboard that is exactly what we are going to do.

Open your Arduino IDE and start a new 'sketch'. Go ahead and save it as 'Tone_Keyboard' in your Arduino folder. Now that it is saved we can add our new header file. In your Arduino IDE go to the Sketch menu and select 'Add File…'. This will open a new dialog box with a lit of files in your sketch's folder. If you right click on the list you can add a new file. Add a new file and name it 'keyboard.h'. **Important Note:** Be sure to remove the existing '.txt' extension. Once you have created the file, double click on 'keyboard.h' and it will load into your Arduino IDE.

This header file will carry all the variables for our Tone Keyboard board. So, first let's go ahead and add the pins for our buttons, LEDs, and speaker.

```
//Assign pins for buttons
int row_1 = 8;
int row_2 = 9;
int row_3 = 3;
int row_4 = 5;
int col_1 = 7;
int col_2 = 4;
int col_3 = 2;
int col_4 = 6;

//Assign pins for LEDs
int led_1 = 10;
int led_2 = 11;
int led_3 = 12;

//Assign pin for speaker
int speaker = 13;
```

Next, we need to create the variables that the ‘Keypad’ library will need to use our buttons. The ‘Keypad’ library creates a keypad object to use. When you create the object you need to use a function called a constructor.

```
Keypad buttons = Keypad(makeKeymap(keys), row[], col[], rows, cols);
```

We are going to walk through this step by step.

`Keypad buttons`	Calls the Keypad object type created by the library and creates an object named ‘buttons’.
`Keypad(`	Calls a ‘Keypad’ function with the library.
`makeKeymap(keys)`	Creates a map for the buttons with names.
`row[]`	List of pins used for rows.
`col[]`	List of pins used for columns.
`Rows`	Count of how many rows there are.
`Cols`	Count of how many columns there are.

Because the button matrix is a grid of four rows and four columns, we need to create variables in our header file to tell the ‘Keypad’ library how to read our button matrix.

```
//List how many rows and columns we have
const byte ROWS = 4;
const byte COLS = 4;

//Connect row and col pins to ‘Keypad’
byte rowPins[ROWS] = {row_1, row_2, row_3, row_4};
byte colPins[COLS] = {col_1, col_2, col_3, col_4};

//Name the different buttons for ‘Keypad’
char keys[ROWS][COLS] = {
   {'0','4','8','c'},
   {'1','5','9','d'},
   {'2','6','a','e'},
   {'3','7','b','f'}
};
```

Great! Next we want to add the tone values for our keyboard. Remember how to use the ‘tone’ function? The ‘tone’ function takes three arguments.

```
tone(speaker_pin, tone_value, tone_duration);
```

We are going to use our header file to store our tone values.

```
//Assign tone values
int C  = 262;
int CS = 277;
int D  = 294;
int DS = 311;
int E  = 330;
int F  = 349;
int FS = 370;
int G  = 392;
int GS = 415;
int A  = 440;
int AS = 466;
int B  = 494;
```

Last, we need to create a list to store the tone values so we can associate the buttons to the tone values.

```
//list of notes
int notes[12] = {
   C, CS,  D, DS,
   E,  F, FS,  G,
  GS,  A, AS,  B
};
```

That’s it! We’re done with our custom header file.

Now, let’s write the main sketch file to control our ‘Tone Keyboard’ kit. The first thing we will need to do is ‘include’ the libraries we want to use. We want to use the ‘keyboard.h’ header we made and the ‘Keypad’ library.

```
#include "keyboard.h"
#include <Keypad.h>
```

Next, create the buttons Keypad object.

```
Keypad buttons = Keypad(makeKeymap(keys),
rowPins, colPins, ROWS, COLS );
```

We also need the values for how long the tone plays for when you press a button.

```
int play_length = 250;
```

Now we need to build our ‘setup()’ function. This means setting up the ‘pinMode’s for the speaker and LEDs.

```
void setup() {
  pinMode(speaker, OUTPUT);
  pinMode(led_1,   OUTPUT);
  pinMode(led_2,   OUTPUT);
  pinMode(led_3,   OUTPUT);
}
```

After this you see the ‘loop()’ function, but we are going to skip it for now. After the ‘loop()’ function we need to create the function ‘playNote()’ that will play the different notes. The function we are going to create will take one argument, the tone value. The center LED will turn on, but only while the tone is playing.

```
void playNote(int note){
  digitalWrite(led_2, HIGH);
  tone(speaker, note, play_length);
  delay(play_length);
  digitalWrite(led_2, LOW);
}
```

Perfect, now we can build our ‘loop()’ function. Remember, the loop function will run over and over. You’ll see the code for the ‘loop()’ function on the next page, but I’m going to over it for you here. In the ‘loop()’ function the first thing we need to do is see if the ‘Keypad’ object sees any buttons being pressed.

```
char key = buttons.getKey();
```

The buttons object has a built-in function called ‘getKey()’. That function checks the button matrix and sees if any of the buttons have been pressed. If a button was pressed it assigned the name of the button to the variable ‘key’. If key has a name attached to it, it checks the name through the ‘if’ statements. If a button matches a name it runs the ‘playNote’ function with the tone value from the ‘notes’ list in our header file. This is where the tone values and buttons names are associated.

After the first twelve buttons, there are the four function keys on the top of the keyboard. I left those if statements empty so that you can add whatever you would like to them. Have it play a short song, light the LEDs, etc.

```
void loop() {
  char key = buttons.getKey();

  if (key){
    if (key == '0'){playNote(notes[0]); }
    if (key == '1'){playNote(notes[1]); }
    if (key == '2'){playNote(notes[2]); }
    if (key == '3'){playNote(notes[3]); }
    if (key == '4'){playNote(notes[4]); }
    if (key == '5'){playNote(notes[5]); }
    if (key == '6'){playNote(notes[6]); }
    if (key == '7'){playNote(notes[7]); }
    if (key == '8'){playNote(notes[8]); }
    if (key == '9'){playNote(notes[9]); }
    if (key == 'a'){playNote(notes[10]);}
    if (key == 'b'){playNote(notes[11]);}

    //FUNCT1
    if (key == 'c'){}
    //FUNCT2
    if (key == 'd'){}
    //FUNCT3
    if (key == 'e'){}
    //FUNCT4
    if (key == 'f'){}
  }
}
```

Go ahead and upload this to your board.

Exercises:

1. Change the ‘playNote’ function to turn on all three LEDs when a note is playing.

2. Make the notes play half as long.

3. Create a function that plays ‘Mary Had a Little Lamb’ when you press the ‘Function 1’ button.

Mary Had a Little Lamb						
GS	FS	E	FS	GS	GS	GS
FS	FS	FS	GS	B	B	
GS	FS	E	FS	GS	GS	GS
GS	FS	FS	GS	FS	E	

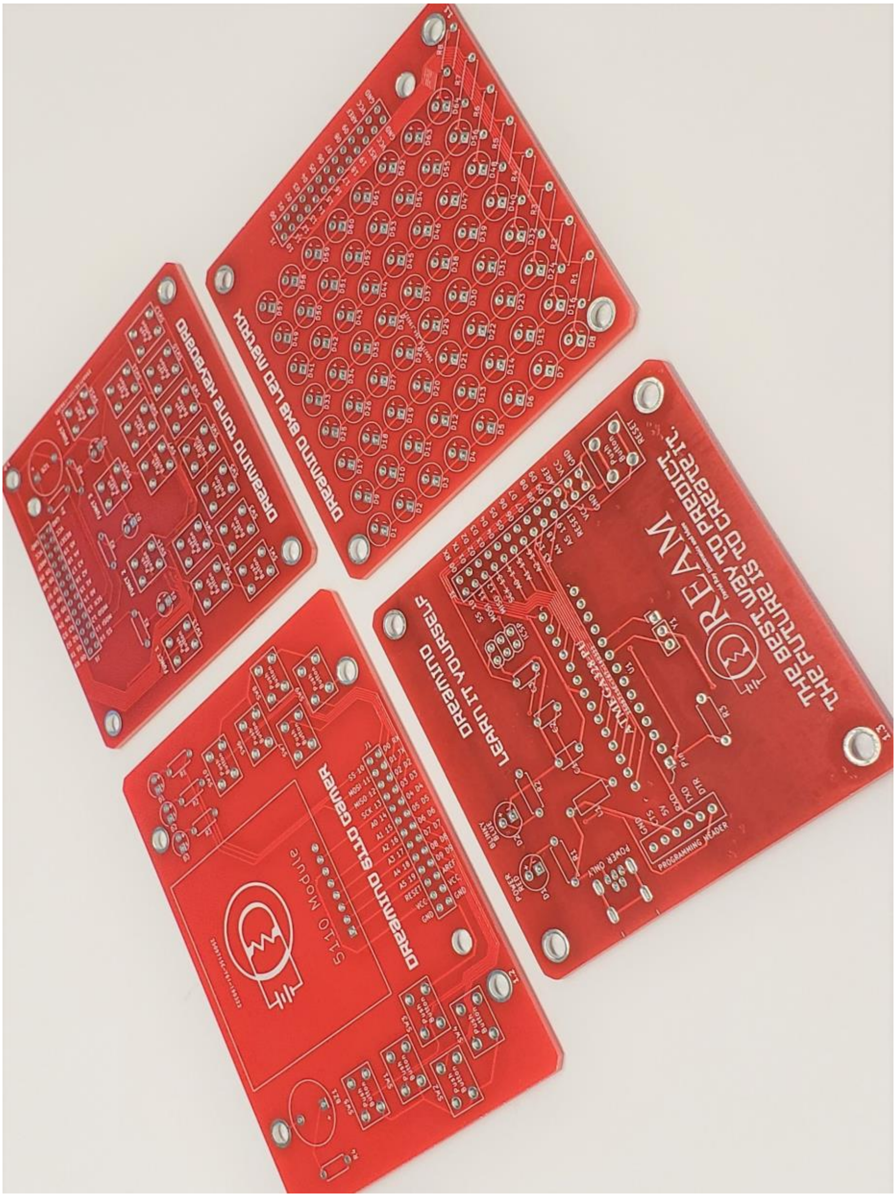
4. Create a function that will cycle the LEDs on and off when you press the ‘Function 4’ button.

Header File: (‘keyboard.h’)

```
//Assign pins for buttons
int row_1 = 8;
int row_2 = 9;
int row_3 = 3;
int row_4 = 5;
int col_1 = 7;
int col_2 = 4;
int col_3 = 2;
int col_4 = 6;

//Assign pins for LEDs
int led_1 = 10;
int led_2 = 11;
int led_3 = 12;

//Assign pin for speaker
int speaker = 13;

//List how many rows and columns we have
const byte ROWS = 4;
const byte COLS = 4;

//Connect row and col pins to ‘Keypad’
byte rowPins[ROWS] = {row_1, row_2, row_3, row_4};
byte colPins[COLS] = {col_1, col_2, col_3, col_4};

//Name the different buttons for ‘Keypad’
char keys[ROWS][COLS] = {
   {'0','4','8','c'},
   {'1','5','9','d'},
   {'2','6','a','e'},
   {'3','7','b','f'}
};

//Assign tone values
int C  = 262;
int CS = 277;
int D  = 294;
int DS = 311;
int E  = 330;
int F  = 349;
int FS = 370;
int G  = 392;
int GS = 415;
int A  = 440;
int AS = 466;
int B  = 494;

//list of notes
int notes[12] = {
   C, CS,  D, DS,
   E,  F, FS,  G,
  GS,  A, AS,  B};
```

Main Sketch File: (‘tone_keyboard.ino’)

```
#include "keyboard.h"
#include <Keypad.h>

Keypad buttons = Keypad(makeKeymap(keys),rowPins,colPins,ROWS,COLS);

int play_length = 250;

void setup() {
  pinMode(speaker, OUTPUT);
  pinMode(led_1,   OUTPUT);
  pinMode(led_2,   OUTPUT);
  pinMode(led_3,   OUTPUT);
}

void playNote(int note){
  digitalWrite(led_2, HIGH);
  tone(speaker, note, play_length);
  delay(play_length);
  digitalWrite(led_2, LOW);
}

void loop() {
  char key = buttons.getKey();

  if (key){
    if (key == '0'){playNote(notes[0]); }
    if (key == '1'){playNote(notes[1]); }
    if (key == '2'){playNote(notes[2]); }
    if (key == '3'){playNote(notes[3]); }
    if (key == '4'){playNote(notes[4]); }
    if (key == '5'){playNote(notes[5]); }
    if (key == '6'){playNote(notes[6]); }
    if (key == '7'){playNote(notes[7]); }
    if (key == '8'){playNote(notes[8]); }
    if (key == '9'){playNote(notes[9]); }
    if (key == 'a'){playNote(notes[10]);}
    if (key == 'b'){playNote(notes[11]);}

    //FUNCT1
    if (key == 'c'){}
    //FUNCT2
    if (key == 'd'){}
    //FUNCT3
    if (key == 'e'){}
    //FUNCT4
    if (key == 'f'){}
  }
}
```


DREAMINO TONE KEYBOARD
DREAMINO 8X8 LED MATRIX
DREAMINO 5110 GAMER
5110 Module
DREAMINO LEARN IT YOURSELF
DREAM
THE BEST WAY TO PREDICT
THE FUTURE IS TO CREATE IT.
ATMEGA328P-PU
PROGRAMMING HEADER
POWER ONLY