**Dawson College**

**Electrical Engineering Technology Department**

**Introduction to Internet of Things**

## **Project Name:**

Musical Box

## **Team Members:**

Anouar Student (ID: 2333814), Syed Student (ID: XXXXXXXX)

May 12, 2025

## 2. Project Description

Meet Jamie, a busy student who loves music but finds traditional instrument frustrating. Between classes and deadlines, they want a quick way to unwind- but sheet music and complex apps that ask for lots of money feel overwhelming.

The solution:

A no-stress music keypad that turns any moment into a musical break. Press buttons to play instant notes or full tunes –no experience needed.

How it works:

-A 4x4 keypad lets users play notes (0-9) or preloaded melodies (A/B buttons)

-A passive buzzer instantly plays prescribed sounds

-Simple design allows one-handed use for quick play sessions

**Final assembly diagram:**

Use any drawing tool of your preference. This is meant to have students think about practical implications for their design. The drawing below was done on PowerPoint. Use the diagram to help with your project description.

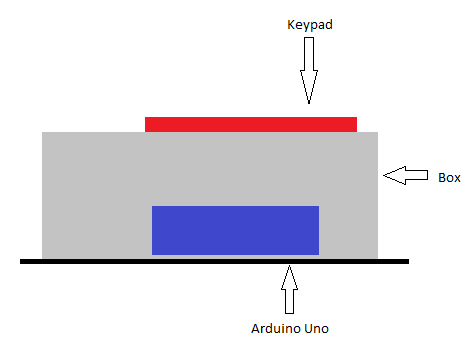


Figure 1 - Project diagram.

## 3. Circuit Diagram

**Inputs:**

* Keypad(4x4)

**Outputs:**

* Passive Buzzer (Plays tones)

Below is a description of the hardware connections, which are also shown graphically using Fritzing (or a similar CAD tool). Note how schematic is **neat, labelled and colour-coded,** so the instructor can follow the circuit at a glance.

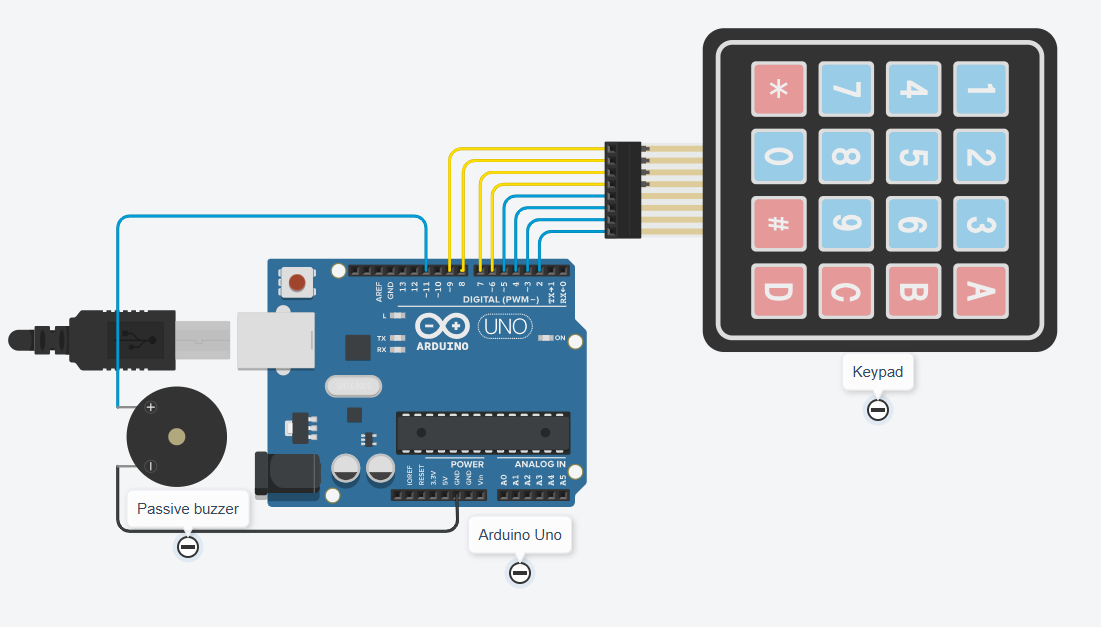


Figure 2 - Circuit Connections Diagram.

**Notes about the circuit assembly:**

1. Notice that the push button (S1) has no pull-up or pull-down resistor. This was intentional as the Arduino Uno allows the pins to be internally connected to pull-up resistors by setting:
   1. pinMode(buttonPin, INPUT\_PULLUP);   // internal pull-up
2. The active buzzer (BZ1) is chosen for its ease of use, only requiring setting it HIGH or LOW to active or deactivate it, respectively.
   1. digitalWrite(buzzerPin, HIGH);
   2. digitalWrite(buzzerPin, LOW);

## 4. Code Documentation

**4.1 Library Used**

**Key pad by mark stanley, alexandre brevin**

**#include <Keypad.h> // Handles keypad input decoding**   
**#include "pitches.h" // Predefined musical note frequencies**   
**#include "Mario.h" // Mario Bros. melody sequence**   
**#include "Pirates\_of\_the\_Caribbeans.h" // Pirates theme melody**

**4.2 Global Constants & Pins**

Below is the list of constant variables used in the code. As constants they do not change during the running of the code, and as global they can be used inside and outside functions.

Table 2 - Global constants and Pins.

|  |  |  |
| --- | --- | --- |
| **Constant / Pin** | **Purpose** | **Typical Value/connection** |
| *buzzerPin* | Passive buzzer | Digital pin 11 |
| *Rows/Cols* | Keypad matrix size | 4 rows,4 columns |
| *Rowpins* | Arduino pins for keypad rows | Pin 9,8,7,6 |
| *colpins* | Arduino pins for keypad columns | Pins 5, 4, 3, 2 |

**4.3 Function Descriptions and Responsibilities**

Table 3 - Function descriptions and Responsibilities.

|  |  |
| --- | --- |
| **Functions** | **Descriptions & Responsibilities** |
| Melody (mario.h) | Plays the Mario Bros. theme: - Uses NOTE\_E5, NOTE\_C5, NOTE\_G5 - Alternates d2 (125ms) and d1 (25ms) delays |
| Melody prate | Plays the Pirates of the Caribbean theme: - Uses NOTE\_A3, NOTE\_C4, NOTE\_D4 - Mixes delay1 (600ms), delay2 (200ms), and delay3 (50ms) |
| Setup() | Initializes hardware: - Sets buzzerPin as output - Starts serial monitor - Tests buzzer with NOTE\_A5 |
| loop() | Main logic: - Checks keypad input with customKeypad.getKey() - Triggers notes/melodies based on pressed keys |

**4.4 Code Explanation**

While the code can be found on our [GitHub](https://github.com/HealthyWaterBottle/IOT_PBL.git)

the explanation of how the code works is shown below.

1. Mario Theme (melodyA() in Mario.h):

Starts with three short E5 notes, followed by a C5 note.

Uses alternating d2 (125 ms) and d1 (25 ms) delays for rhythmic pacing.

Example:

tone(buzzerPin, NOTE\_E5, 125); delay(125); // First "Mario" beep   
noTone(buzzerPin); delay(25); // Short pause

1. **(melodyB() in Pirates\_of\_the\_Caribbeans.h):**

Begins with a slow A3-C4-D4 sequence.

Uses three delay intervals (delay1=600ms, delay2=200ms, delay3=50ms) for dramatic pacing.

Example:

tone(buzzerPin, NOTE\_A3, 200); delay(200); // Opening "Pirates" note   
noTone(buzzerPin); delay(50); // Micro-pause

1. pitches.h

Purpose: Defines all note frequencies (e.g., NOTE\_C4 = 262 Hz).

Required: Without this file, the code will not compile (notes like NOTE\_A5 would be undefined).

Usage: Included in the main sketch and melody files (Mario.h, Pirates.h).

1. Mechanism: The loop() continuously calls customKeypad.getKey(), which scans the 4x4 matrix for pressed keys.

Output:

Detected keys are printed to the Serial Monitor (e.g., Key pressed: A).

Number keys (0-9) trigger single notes.

A/B keys trigger full melodies.

1. Buzzer Control

A passive buzzer is used, requiring tone() for frequency control.

noTone(buzzerPin) ensures clean note separation.

1. The loop() function continuously checks for user input and coordinates all system responses. :

Detects key presses from the keypad.

Logs inputs to the Serial Monitor for debugging.

Triggers melodies or notes based on the pressed key.

Maintains responsiveness during playback (except during melodies).

Together these steps create musical beat on a click: by pressing any key on pad loads the associated tune given by the code that sends an output given frequency.

## 5. Ethics, Privacy, or Security Disclaimer

-Ethical Design: This project is for entertainment/education only. It does not collect user data.

-Component Choice: Passive buzzers consume less power than active buzzers.

-Reusability: Code can be adapted for other melodies by editing pitches.h and melody files.

-Safety: Low-voltage design (5V) poses no electrical risk.

## 6. References

Mario theme: <https://www.youtube.com/watch?v=7i3oM-u93xM>

Pirates Of The Caribbean: <https://www.youtube.com/watch?v=DmpJfK70dgg>

Keypad integration for music input and Pitches.h: <https://spot.pcc.edu/~dgoldman/labs/37SENSORKIT.pdf>

Arduino uno tone use: <https://docs.arduino.cc/language-reference/en/functions/advanced-io/tone/>