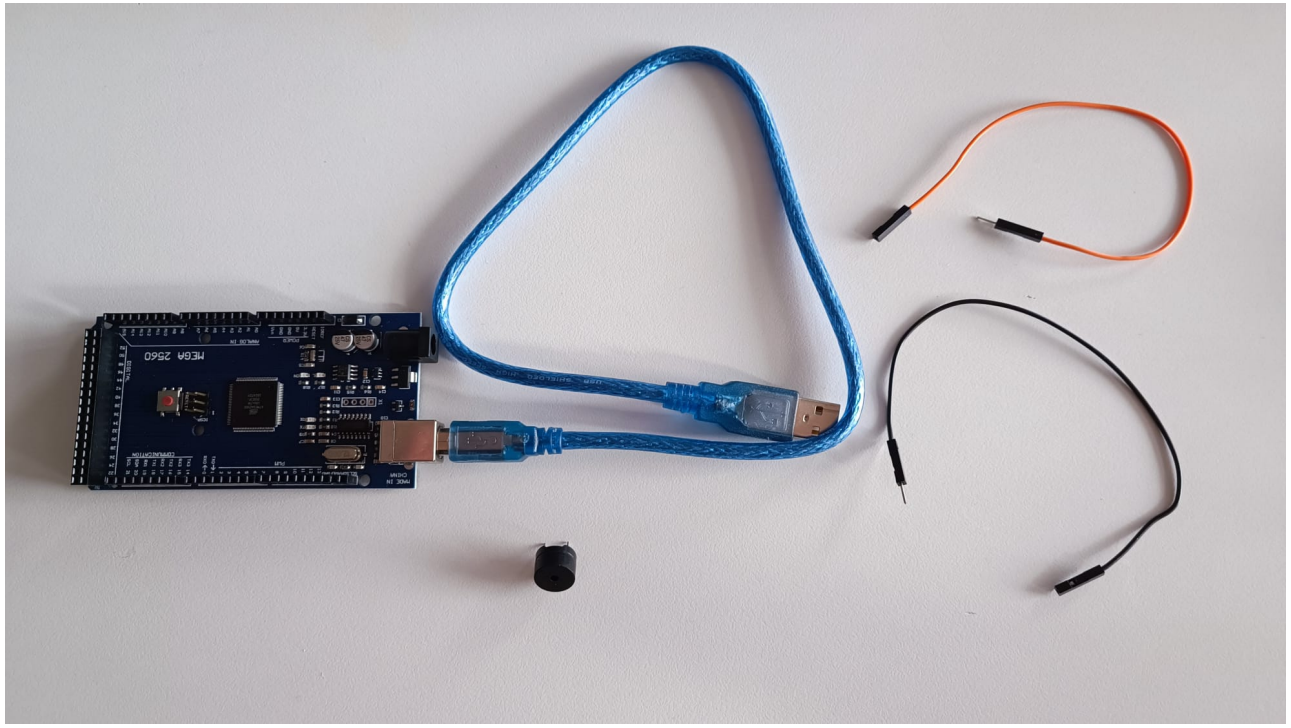
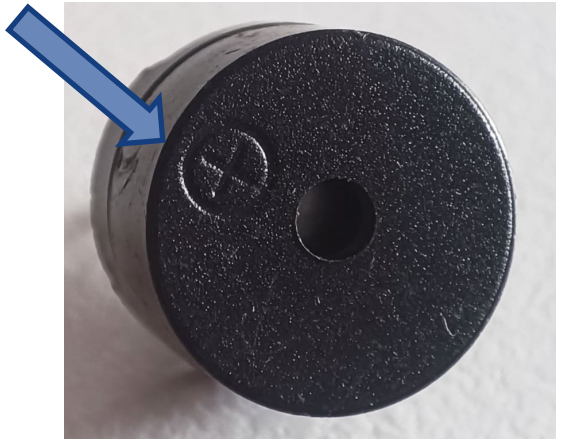
Proyecto Arduino – Zumbador pasivo

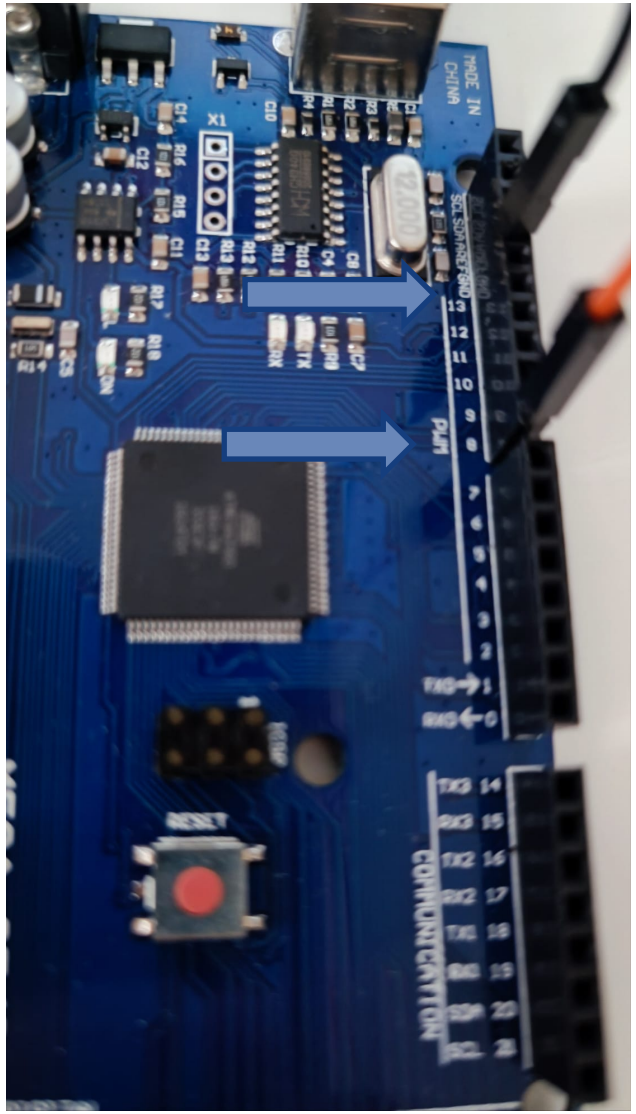
- Componentes necesarios para el proyecto:
  - Arduino
  - Zumbador pasivo (passive buzzer)
  - 2 cables (macho-hembra) para la conexión del zumbador





Proyecto Arduino – Zumbador pasivo

- Conexión
  - 1 Cable va del positivo del Zumbador al PIN 8
  - 2do Cable va del negativo del Zumbador a GND (tierra)



Proyecto Arduino – Zumbador pasivo

- Programación
  - **Definición de las notas**
  - Asignación de los pines
  - Nota musical (canción)
  - Duración

Datei Bearbeiten Sketch Werkzeuge Hilfe

Hochladen

20191013_Buzzer_Passive §

```
// Defintion der Noten Frequenzen
// Definición de Notas y Frecuencias
#define DOO   523
#define RE    294
#define MI    330
#define FA    349
#define SO    392
#define LA    440
#define SI    494
#define doo   523

// Pinbelegung
// Asignación de los PINs
const uint8_t BUZZER = 8;
const uint8_t LED1 = 13;

// Noten (Tonhöhe)
// Notas musicales
int notes[] =
{
   DOO, RE, MI, FA, 0,
   SO, LA, SI, doo, 0,

};

// Notenwerte (Tondauer)
// Um die Tondauer zu berechnen,
// nehmen eine Sekunde, geteilt durch den Notentyp
// z.B. viertel Note = 1000 / 4, Achtel Note = 1000/8, etc.
// Duración de las notas

int duration[] =
{
  300, 300, 300, 300, 700,
  300, 300, 300, 300, 700,

};
```

Proyecto Arduino – Zumbador pasivo

- Programación
  - Definición de las notas
  - **Asignación de los pines**
  - Nota musical (canción)
  - Duración

Datei Bearbeiten Sketch Werkzeuge Hilfe

Hochladen

20191013_Buzzer_Passive §

```
// Defintion der Noten Frequenzen
// Definición de Notas y Frecuencias
#define DOO   523
#define RE    294
#define MI    330
#define FA    349
#define SO    392
#define LA    440
#define SI    494
#define doo   523

// Pinbelegung
// Asignación de los PINs
const uint8_t BUZZER = 8;
const uint8_t LED1 = 13;

// Noten (Tonhöhe)
// Notas musicales
int notes[] =
{
   DOO, RE, MI, FA, 0,
   SO, LA, SI, doo, 0,

};

// Notenwerte (Tondauer)
// Um die Tondauer zu berechnen,
// nehmen eine Sekunde, geteilt durch den Notentyp
// z.B. viertel Note = 1000 / 4, Achtel Note = 1000/8, etc.
// Duración de las notas

int duration[] =
{
  300, 300, 300, 300, 700,
  300, 300, 300, 300, 700,

};
```

Proyecto Arduino – Zumbador pasivo

- Programación
  - Definición de las notas
  - Asignación de los pines
  - **Nota musical (canción)**
  - Duración

Datei Bearbeiten Sketch Werkzeuge Hilfe

Hochladen

20191013_Buzzer_Passive §

```
// Defintion der Noten Frequenzen
// Definición de Notas y Frecuencias
#define DOO   523
#define RE    294
#define MI    330
#define FA    349
#define SO    392
#define LA    440
#define SI    494
#define doo   523

// Pinbelegung
// Asignación de los PINs
const uint8_t BUZZER = 8;
const uint8_t LED1 = 13;

// Noten (Tonhöhe)
// Notas musicales
int notes[] =
{
   DOO, RE, MI, FA, 0,
   SO, LA, SI, doo, 0,

};

// Notenwerte (Tondauer)
// Um die Tondauer zu berechnen,
// nehmen eine Sekunde, geteilt durch den Notentyp
// z.B. viertel Note = 1000 / 4, Achtel Note = 1000/8, etc.
// Duración de las notas

int duration[] =
{
  300, 300, 300, 300, 700,
  300, 300, 300, 300, 700,

};
```

Proyecto Arduino – Zumbador pasivo

- Programación
  - Definición de las notas
  - Asignación de los pines
  - Nota musical (canción)
  - **Duración**

Datei Bearbeiten Sketch Werkzeuge Hilfe

Hochladen

20191013_Buzzer_Passive §

```
// Defintion der Noten Frequenzen
// Definición de Notas y Frecuencias
#define DOO   523
#define RE    294
#define MI    330
#define FA    349
#define SO    392
#define LA    440
#define SI    494
#define doo   523

// Pinbelegung
// Asignación de los PINs
const uint8_t BUZZER = 8;
const uint8_t LED1 = 13;

// Noten (Tonhöhe)
// Notas musicales
int notes[] =
{
   DOO, RE, MI, FA, 0,
   SO, LA, SI, doo, 0,

};

// Notenwerte (Tondauer)
// Um die Tondauer zu berechnen,
// nehmen eine Sekunde, geteilt durch den Notentyp
// z.B. viertel Note = 1000 / 4, Achtel Note = 1000/8, etc.
// Duración de las notas
```

```
int duration[] =
{
  300, 300, 300, 300, 700,
  300, 300, 300, 300, 700,

};
```

Proyecto Arduino – Zumbador pasivo

- Programación detallada
  - **Asignación de encendido/apagado**
  - Asignación de notas vs duración
  - Matriz

```
void setup() {
  // I/O-Pin als Ausgang
  pinMode(LED1, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  digitalWrite(LED1, HIGH);   // LED aus
  digitalWrite(BUZZER, HIGH); // BUZZER aus (Low-Aktiv)
}
```

```
void loop() {
    int size = sizeof(notes) / sizeof(int);
    for (int thisNote = 0; thisNote < size ; thisNote++) {
      int noteDuration = duration[thisNote];

      buzz(notes[thisNote], noteDuration);

      // Um die Noten zu unterscheiden, wird eine Mindestze:
      int pauseBetweenNotes = noteDuration * 0.5;
      delay(pauseBetweenNotes);
    }
}

void buzz(long frequency, long length) {
  digitalWrite(LED1, LOW); // LED AN
  long delayValue = 1000000 / frequency / 2; // Berechnung
  // 1 Sekunde in Mikrosekunden, geteilt durch die Frequenz
  long numCycles = frequency * length / 1000; // Berechnung
  // Frequenz, die die wirklichen Zyklen pro Sekunde ist, m
    for (long i = 0; i < numCycles; i++) { // for the calcu
    digitalWrite(BUZZER, LOW); // BUZZER an
    delayMicroseconds(delayValue); // warten auf den berechn
    digitalWrite(BUZZER, HIGH); // BUZZER aus
    delayMicroseconds(delayValue); // warten auf den berechn
  }
  digitalWrite(LED1, HIGH); // LED AUS
}
```

Proyecto Arduino – Zumbador pasivo

- Programación detallada
  - Asignación de encendido/apagado
  - **Asignación de notas vs duración**
  - Matriz

```
void setup() {
  // I/O-Pin als Ausgang
  pinMode(LED1, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  digitalWrite(LED1, HIGH);   // LED aus
  digitalWrite(BUZZER, HIGH); // BUZZER aus (Low-Aktiv)
}
```

```
void loop() {
    int size = sizeof(notes) / sizeof(int);
    for (int thisNote = 0; thisNote < size ; thisNote++) {
      int noteDuration = duration[thisNote];

      buzz(notes[thisNote], noteDuration);

      // Um die Noten zu unterscheiden, wird eine Mindestze
      int pauseBetweenNotes = noteDuration * 0.5;
      delay(pauseBetweenNotes);
    }
}
```

```
void buzz(long frequency, long length) {
  digitalWrite(LED1, LOW); // LED AN
  long delayValue = 1000000 / frequency / 2; // Berechnung
  // 1 Sekunde in Mikrosekunden, geteilt durch die Frequenz
  long numCycles = frequency * length / 1000; // Berechnung
  // Frequenz, die die wirklichen Zyklen pro Sekunde ist, m
    for (long i = 0; i < numCycles; i++) { // for the calcu
    digitalWrite(BUZZER, LOW); // BUZZER an
    delayMicroseconds(delayValue); // warten auf den berech
    digitalWrite(BUZZER, HIGH); // BUZZER aus
    delayMicroseconds(delayValue); // warten auf den berech
  }
  digitalWrite(LED1, HIGH); // LED AUS
}
```

Proyecto Arduino – Zumbador pasivo

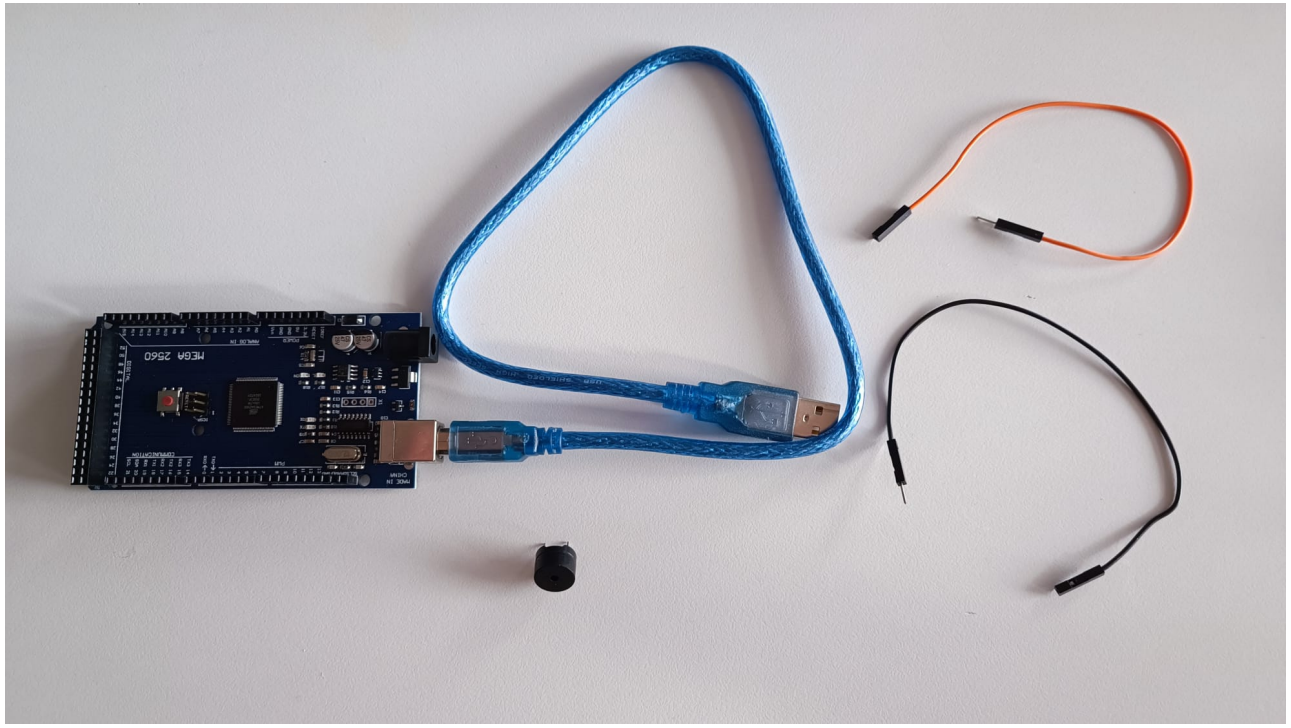
- Programación detallada
  - Asignación de encendido/apagado
  - Asignación de notas vs duración
  - **Matriz**

```
void setup() {
  // I/O-Pin als Ausgang
  pinMode(LED1, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  digitalWrite(LED1, HIGH);   // LED aus
  digitalWrite(BUZZER, HIGH); // BUZZER aus (Low-Aktiv)
}

void loop() {
    int size = sizeof(notes) / sizeof(int);
    for (int thisNote = 0; thisNote < size ; thisNote++) {
      int noteDuration = duration[thisNote];

      buzz(notes[thisNote], noteDuration);

      // Um die Noten zu unterscheiden, wird eine Mindestze:
      int pauseBetweenNotes = noteDuration * 0.5;
      delay(pauseBetweenNotes);
    }
}
```

```
void buzz(long frequency, long length) {
  digitalWrite(LED1, LOW); // LED AN
  long delayValue = 1000000 / frequency / 2; // Berechnung d
  // 1 Sekunde in Mikrosekunden, geteilt durch die Frequenz
  long numCycles = frequency * length / 1000; // Berechnung
  // Frequenz, die die wirklichen Zyklen pro Sekunde ist, m
    for (long i = 0; i < numCycles; i++) { // for the calcu
    digitalWrite(BUZZER, LOW); // BUZZER an
    delayMicroseconds(delayValue); // warten auf den berechn
    digitalWrite(BUZZER, HIGH); // BUZZER aus
    delayMicroseconds(delayValue); // warten auf den berechn
  }
  digitalWrite(LED1, HIGH); // LED AUS
}
```

Proyecto Arduino – Zumbador pasivo

- Código
  - Si tuviste problemas en programar, en las siguientes páginas puedes copiar el código.



Principios y fundamentos

- Código
  - // Defintion der Noten Frequenzen
  - #define DOO 523
  - #define RE 294
  - #define MI 330
  - #define FA 349
  - #define SO 392
  - #define LA 440
  - #define SI 494
  - #define doo 523

  - // Pinbelegung
  - const uint8_t BUZZER = 8;
  - const uint8_t LED1 = 13;

  - // Noten (Tonhöhe)
  - int notes[] =
  - {
  - DOO, RE, MI, FA, 0,
  - SO, LA, SI, doo, 0,
  -
  - };

  - // Notenwerte (Tondauer)
  - // Um die Tondauer zu berechnen,
  - // nehmen eine Sekunde, geteilt durch den Notentyp
  - // z.B. viertel Note = 1000 / 4, Achtel Note = 1000/8, etc.

  - int duration[] =
  - {
  - 300, 300, 300, 300, 700,
  - 300, 300, 300, 300, 700,
  -
  - };

Principios y fundamentos

- Código

```
void setup() {
  // I/O-Pin als Ausgang
  pinMode(LED1, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  digitalWrite(LED1, HIGH);   // LED aus
  digitalWrite(BUZZER, HIGH); // BUZZER aus (Low-Aktiv)
}

void loop() {
    int size = sizeof(notes) / sizeof(int);
    for (int thisNote = 0; thisNote < size ; thisNote++) {
      int noteDuration = duration[thisNote];

      buzz(notes[thisNote], noteDuration);

      // Um die Noten zu unterscheiden, wird eine Mindestzeit zwischen ihnen festgelegt.
      int pauseBetweenNotes = noteDuration * 0.5;
      delay(pauseBetweenNotes);
    }
}

void buzz(long frequency, long length) {
  digitalWrite(LED1, LOW); // LED AN
  long delayValue = 1000000 / frequency / 2; // Berechnung des Verzögerungswertes zwischen den Übergängen
  // 1 Sekunde in Mikrosekunden, geteilt durch die Frequenz und dann halbiert, da zu jedem Zyklus zwei Phasen vorhanden sind
  long numCycles = frequency * length / 1000; // Berechnung der Anzahl der Zyklen für das richtige Timing
  // Frequenz, die die wirklichen Zyklen pro Sekunde ist, multipliziert mit der Anzahl der Sekunden,um die Gesamtzahl der Zyklen zu erhalten
    for (long i = 0; i < numCycles; i++) { // for the calculated length of time...
    digitalWrite(BUZZER, LOW); // BUZZER an
    delayMicroseconds(delayValue); // warten auf den berechneten Verzögerungswert
    digitalWrite(BUZZER, HIGH); // BUZZER aus
    delayMicroseconds(delayValue); // warten auf den berechneten Verzögerungswert
  }
  digitalWrite(LED1, HIGH); // LED AUS
}
```