Project: Uselesser Machine

Robotics
ME 5493-001
Pranav Bhounsule

Edward A. Young
Hugo Vasquez
Melvin R. Ayuso

12/17/2017

Abstract

A useless machine was built in a way that makes it more useless than the simple useless box. The project contains a small “desk” with a switch that gives a signal to the microcontroller which then controls the relay. The light can be programmed to flash at every beat of the song, or can also be programmed to turn off after ‘x’ amount of seconds. ‘X’ is a number chosen and input by the user programming the machine. A buzzer plays the melody of the song when the switch is turned on, which is also controlled by the microcontroller. The electronics assembly was placed inside a MDF box that was built by the team. This desk lamp is a uselesser machine than the original machine. The team plans on continuing work on the project and adding more useless features to the useless “desk”.

Introduction

The microcontroller can be used for a multitude of purposes when paired with different sensors, including using a moisture sensor to create an automatic plant watering device or using laser LEDs and photoresistors to create a laser security grid. For this project, it will be utilized in such a way that it will help make useful things such as a desk lamp becoming not so useful anymore. The initial design was to just make a lamp turn off and on. Later, the buzzer was integrated with the initial design to make the lamp turn on and off in synchronization with the beat of a melody coming from the buzzer. Figure 1 shows the electrical schematic of the design.

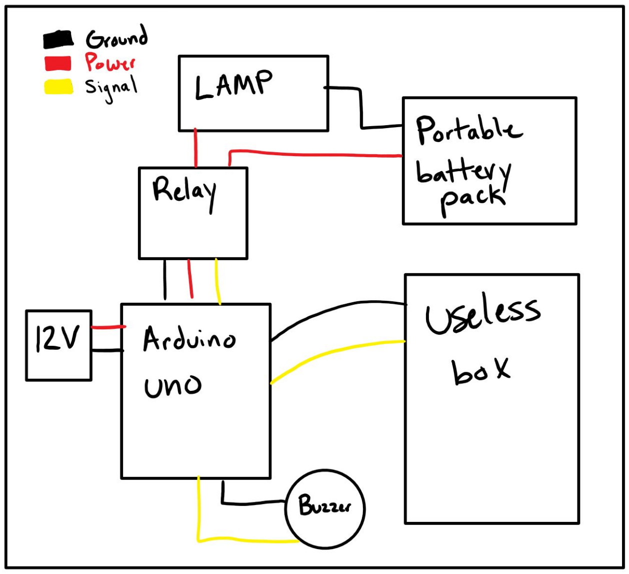

Figure 1: Electrical Schematic

Setup/Build

The team began by brainstorming as to what could be integrated into the useless machine to make it uselesser. While speaking with Professor Bhonsoule, he suggested integrating a lamp or putting the useless machine into a flashlight or something along those lines. The team chose to go with the lamp idea. The initial plan was to install the lamp onto the box of the useless machine, but after looking at the sizes of the boxes that the useless machines, it was determined that this would not be possible. The plan was then switched to now build a larger box and install both the lamp and useless machine into there. This would also leave room for further expansion of the uselesser machine's capabilities.

A box was made out of ¼" MDF (medium density fiberboard). The MDF board was purchased at Homedepot and was initially attempted to be cut to size with a dremel, however it was immediately apparent that this would take longer than expected. The team switched to a circular saw and proceeded to cut two pieces, each measuring 15 inches by 15 inches. These pieces made up the top and bottom of the desk. Afterwards, four pieces measuring approximately 2" in width were cut out for the sidewalls of the box. After all the mdf pieces were cut out, wood glue and small finishing nails were used to put the box together. The top piece of the board had a small cut out so the useless box would be able to fit flushed with the MDF board ,and so that the MDF board would not interfere with the useless box's operation. The figure below shows some of the build process.

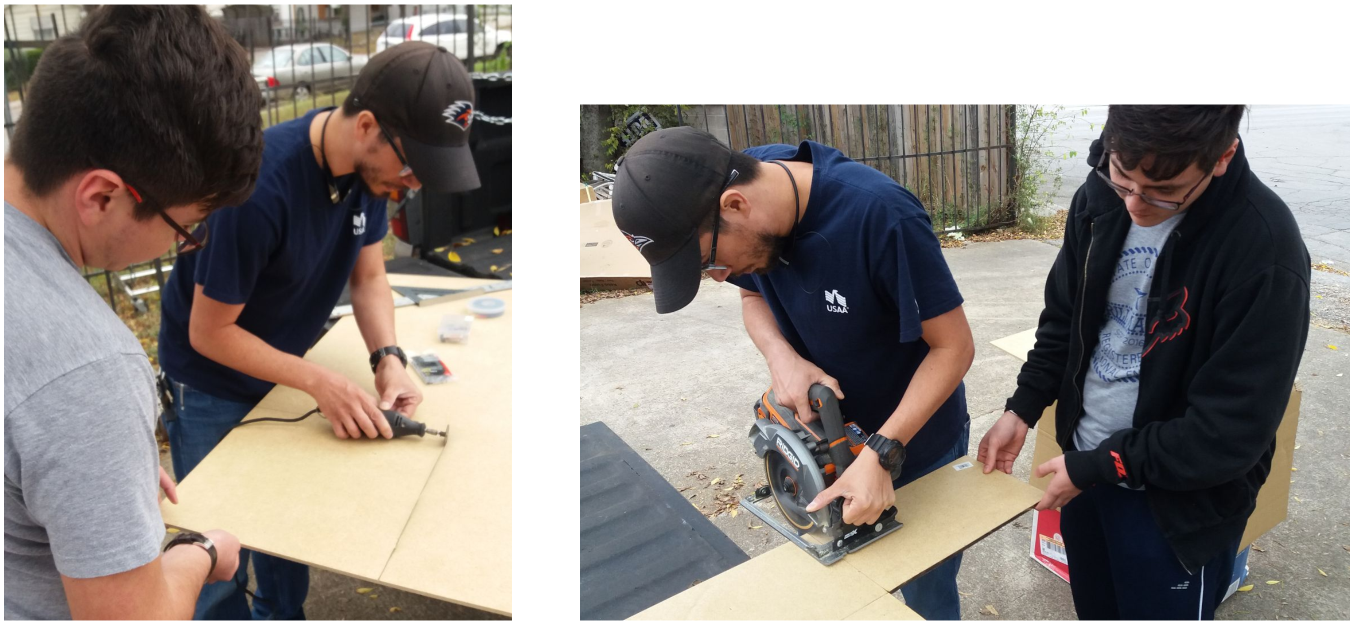

Figure 2: Cutting MDF box housing with a Dremel (Left) Cutting using a circular saw (Right)

After the box was fully built, the wiring began. The useless box had a toggle switch to provide an H-bridge to manage changing the direction of the small DC motor that it housed. This means that when the switch is turned on, the motor spins the opposite direction so that it travels upward until the switch is turned off. It then reverses direction and travels until until it contacts the limiter switch, which breaks the circuit thereby cutting off power to the motor. Two pins were branched off from so that a 3 volt signal was going into the microcontroller as shown in figure 3.

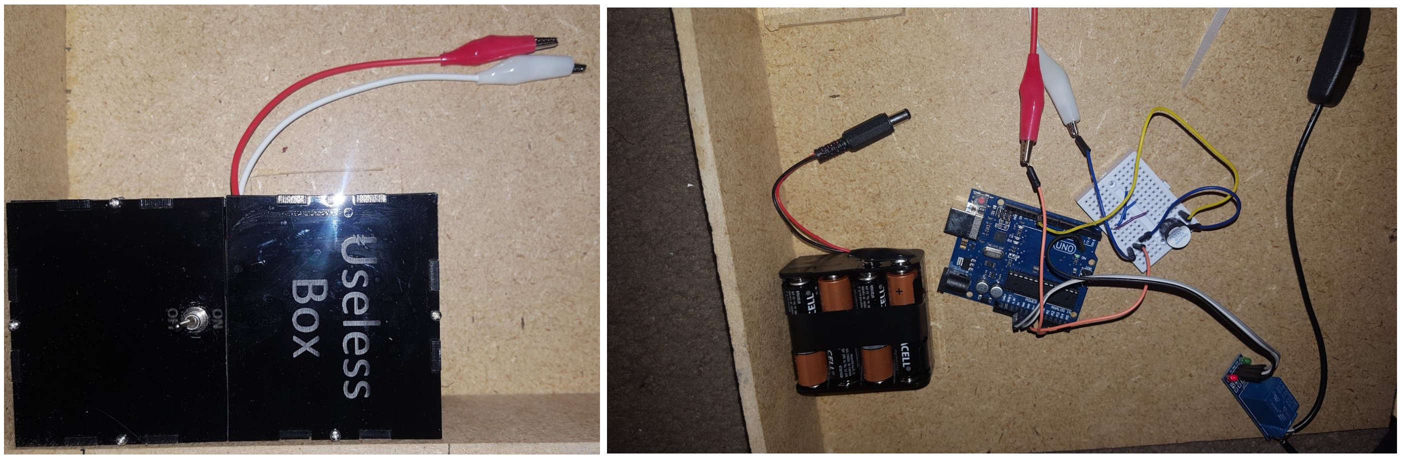

Figure 3: Wires branching off from H-bridge (Left) Connected to Arduino Uno (Right)

When the switch is turned on, the voltage drops approximately to 2.4 volts due to the voltage drop across the DC motor. For the coding portion, when the voltage drops less than 2.8 volts, the microcontroller will perform the assigned tasks of switching the relay on and off to control the desk lamp and powering the pin to sound the buzzer. Three pins were used to connect to the relay, one each providing power, ground, and signal out from the arduino and into the relay. The signal acts like an off/on switch, being that when a 5 volt signal is provided the circuit is open or “on”, and when a 0 volt signal, or ground, is provided the circuit is closed or “off”. Figure 4 shows the relay that was used for this project. The side of the relay with the screw terminals is what interrupts the “hot” line of the plug and gives power to the lamp.

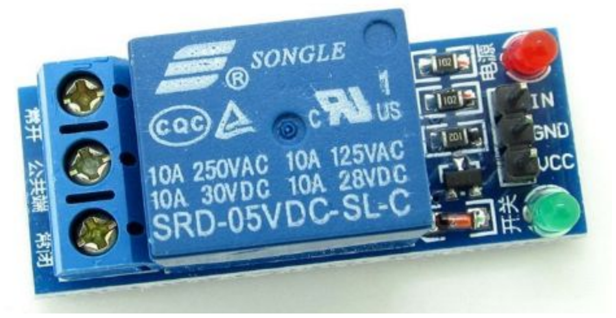

Figure 4: Relay

The figure below shows how the wiring is physically connected. To elaborate a little on how the lamp’s power is controlled and provided, the bottom is where the black cables from the lamp’s plug are connected to the relay. The wire sheathing was stripped off, and the “hot” wire was cut and then connected to the relay. The other wires are what goes to the arduino. The black, white, and grey wires are the signal, ground, and power (5 VDC) respectively.

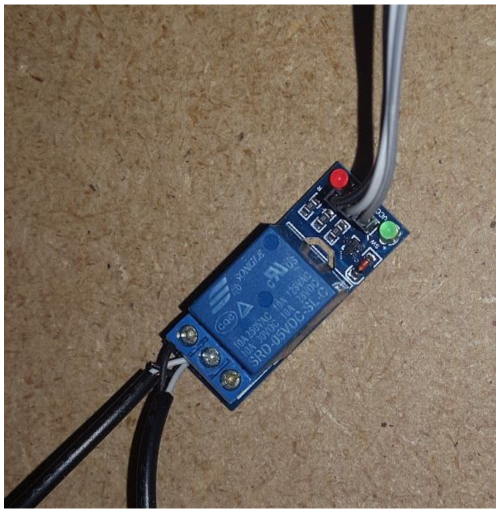

Figure 5: Relay with all wire connections

The way the lamp originally gets power is by a USB cord and small power transformer box that connects to an outlet plug. To make the project portable, or operate without needing outlet plugs, a 10,000mAh rechargeable portable battery pack was used. This charger has a USB output of 5 volts and 1 amp. This was determined to be sufficient enough to power the lamp and can be seen in figure 6.

Figure 6: 10,000 mAh Portable Battery Pack with USB lamp connection

The completed physical wiring can be found in figure 7 and is similar to what was shown in figure 1. While running the code, a picture was taken to show that the green led light turns on indicating a low voltage was detected thus creating a closed circuit for the lamp. This can be seen also in figure 7. This is when the buzzer is turned on and plays the melody [1]. The code is attached in the appendix section. The melody being played was not created by the team. Several melodies were found on open source code sharing sites and experimented with, but some were more difficult to get to function with the relay and lamp. The coded melody was integrated to work with the code created by the team to control the relay on and off. Essentially, the melody of the song dictates how the lamp will turn on and off and turns on and off in accordance with the beat the song being played. The completed design is found in figure 8.

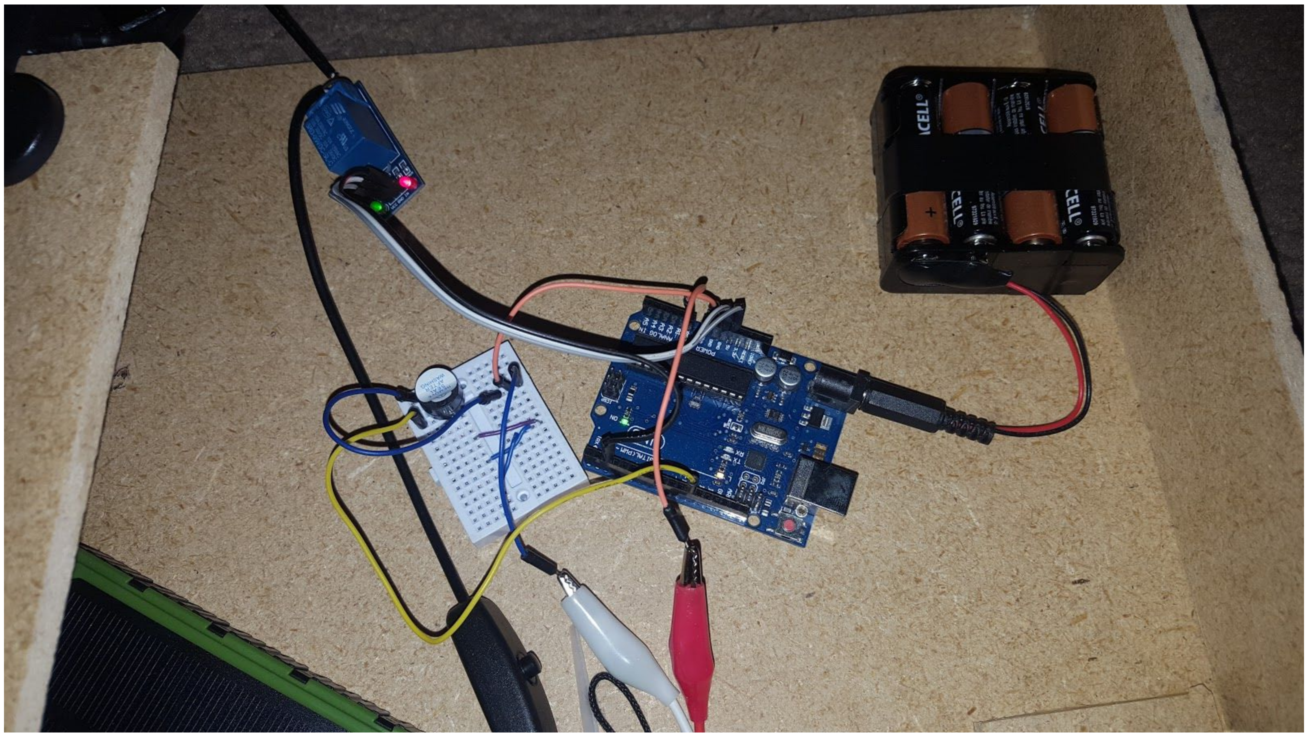

Figure 7: Complete setup (green light showing relay circuit is closed)

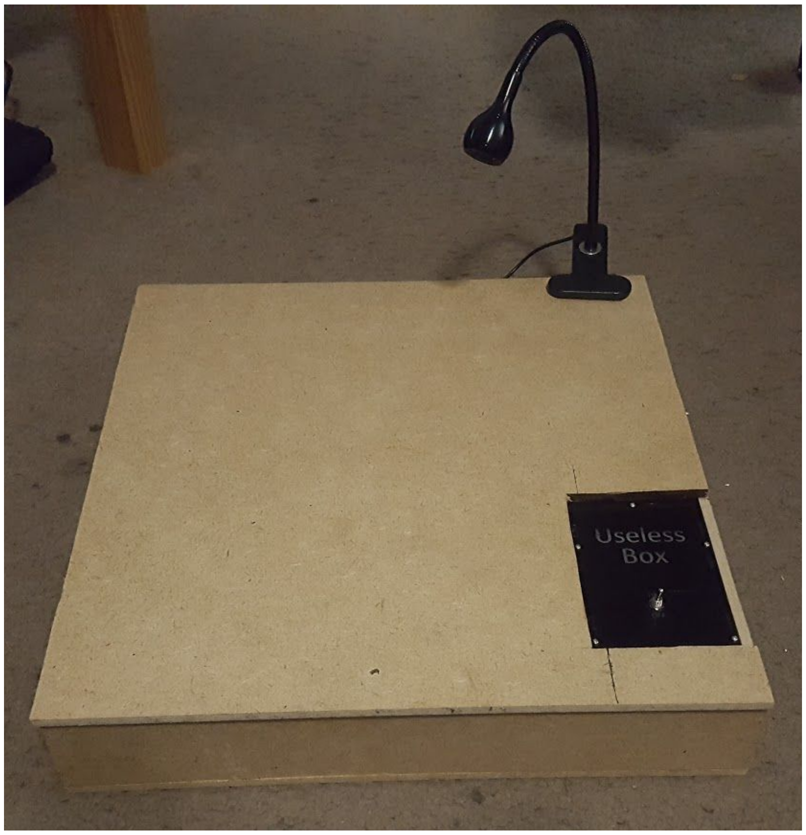

Figure 8: Completed Project Build

Lessons Learned

1. The relay being used was different from what the team was used to working with. It required a 0V “low” pin trigger as opposed to a 5V “high” pin trigger.
2. When using an arduino nano, the voltage of the battery pack creates some interference, thereby affecting the readings from the useless box.
3. The Dremel was helpful, but took way too much time to use during fabrication due to the size of the tool. Other tools, such as a bandsaw or jigsaw, would have been more appropriate for this project’s fabrication.

Conclusion

The uselesser machine is a very useless machine. The team was able to successfully produce a uselesser machine. The arduino itself has more potential for this project. The team will continue working on this project, coming up with unique ways to make it even more useless and frustrating for the user. The user may or may not have any clue that this is useless since this could be given as a “gift”. For testing, we hope to get individuals that are unsuspecting to test the uselessness of it. Overall, the team is happy with the results so far, and the individuals who have helped to test it so far have enjoyed the project. Thank you for taking the time to read this.

Reference

[1] https://www.princetronics.com/supermariothemesong/

Appendix

```
// the setup routine runs once when you press reset:
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
```

```
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978

#define melodyPin 11
//Mario main theme melody
```

```
int melody[] = {
  NOTE_E7, NOTE_E7, 0, NOTE_E7,
  0, NOTE_C7, NOTE_E7, 0,
  NOTE_G7, 0, 0,  0,
  NOTE_G6, 0, 0, 0,

  NOTE_C7, 0, 0, NOTE_G6,
  0, 0, NOTE_E6, 0,
  0, NOTE_A6, 0, NOTE_B6,
  0, NOTE_AS6, NOTE_A6, 0,

  NOTE_G6, NOTE_E7, NOTE_G7,
  NOTE_A7, 0, NOTE_F7, NOTE_G7,
  0, NOTE_E7, 0, NOTE_C7,
  NOTE_D7, NOTE_B6, 0, 0,

  NOTE_C7, 0, 0, NOTE_G6,
  0, 0, NOTE_E6, 0,
  0, NOTE_A6, 0, NOTE_B6,
  0, NOTE_AS6, NOTE_A6, 0,

  NOTE_G6, NOTE_E7, NOTE_G7,
  NOTE_A7, 0, NOTE_F7, NOTE_G7,
  0, NOTE_E7, 0, NOTE_C7,
  NOTE_D7, NOTE_B6, 0, 0
};
//Mario main them tempo
int tempo[] = {
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,

  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,

  9, 9, 9,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,

  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  9, 9, 9,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
};
//Underworld melody
int underworld_melody[] = {
  NOTE_C4, NOTE_C5, NOTE_A3, NOTE_A4,
  NOTE_AS3, NOTE_AS4, 0,
  0,
  NOTE_C4, NOTE_C5, NOTE_A3, NOTE_A4,
  NOTE_AS3, NOTE_AS4, 0,
  0,
  NOTE_F3, NOTE_F4, NOTE_D3, NOTE_D4,
  NOTE_DS3, NOTE_DS4, 0,
  0,
  NOTE_F3, NOTE_F4, NOTE_D3, NOTE_D4,
  NOTE_DS3, NOTE_DS4, 0,
  0, NOTE_DS4, NOTE_CS4, NOTE_D4,
  NOTE_CS4, NOTE_DS4,
  NOTE_DS4, NOTE_GS3,
  NOTE_G3, NOTE_CS4,
  NOTE_C4, NOTE_FS4, NOTE_F4, NOTE_E3, NOTE_AS4, NOTE_A4,
  NOTE_GS4, NOTE_DS4, NOTE_B3,
  NOTE_AS3, NOTE_A3, NOTE_GS3,
  0, 0, 0
};
//Underwolrd tempo
int underworld_tempo[] = {
  12, 12, 12, 12,
  12, 12, 6,
  3,
  12, 12, 12, 12,
  12, 12, 6,
  3,
  12, 12, 12, 12,
  12, 12, 6,
  3,
  12, 12, 12, 12,
  12, 12, 6,
  6, 18, 18, 18,
  6, 6,
  6, 6,
  6, 6,
  18, 18, 18, 18, 18, 18,
  10, 10, 10,
  10, 10, 10,
  3, 3, 3
```

```
};
int n=100; //number of samples per second
int b=1.5; // the delay in seconds for light to be on
int relay=2;
int buzzer=11;
int i=0;

void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  Serial.print("voltage");
  pinMode(relay,OUTPUT);
  pinMode(buzzer, OUTPUT);//buzzer


}

// the loop routine runs over and over again forever:
void loop() {

  // read the input on analog pin 0:
  int sensorValue = analogRead(0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (4.6 / 1023.0);

  Serial.println(voltage);

  delay(1000/n);
  if (voltage > 2.8)
  {
  digitalWrite(relay,HIGH);
 // delay(1000*b);
  }

  else if (voltage <2.8)
  {
  digitalWrite(relay,LOW);
  delay(1000*b);
  i=i+1;
  sing(i);
  }
  if (i==2){
    i=0;
    }
}
int song = 0;

void sing(int s) {
  // iterate over the notes of the melody:
  song = s;
  if (song == 2) {
    Serial.println(" 'Underworld Theme'");
    int size = sizeof(underworld_melody) / sizeof(int);
    for (int thisNote = 0; thisNote < size; thisNote++) {

      // to calculate the note duration, take one second
      // divided by the note type.
      //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
      int noteDuration = 1000 / underworld_tempo[thisNote];

      buzz(melodyPin, underworld_melody[thisNote], noteDuration);

      // to distinguish the notes, set a minimum time between them.
      // the note's duration + 30% seems to work well:
      int pauseBetweenNotes = noteDuration * 1.30;
      delay(pauseBetweenNotes);

      // stop the tone playing:
      buzz(melodyPin, 0, noteDuration);

    }

  } else {

   // Serial.println(" 'Mario Theme'");
    int size = sizeof(melody) / sizeof(int);
    for (int thisNote = 0; thisNote < size; thisNote++) {

      // to calculate the note duration, take one second
      // divided by the note type.
      //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
      int noteDuration = 1000 / tempo[thisNote];

      buzz(melodyPin, melody[thisNote], noteDuration);

      // to distinguish the notes, set a minimum time between them.
      // the note's duration + 30% seems to work well:
      int pauseBetweenNotes = noteDuration * 1.30;
      delay(pauseBetweenNotes);

      // stop the tone playing:
      buzz(melodyPin, 0, noteDuration);

    }
  }
}
```

```
void buzz(int targetPin, long frequency, long length) {
  digitalWrite(2, HIGH);
  long delayValue = 1000000 / frequency / 2; // calculate the delay value between transitions
  //// 1 second's worth of microseconds, divided by the frequency, then split in half since
  //// there are two phases to each cycle
  long numCycles = frequency * length / 1000; // calculate the number of cycles for proper timing
  //// multiply frequency, which is really cycles per second, by the number of seconds to
  //// get the total number of cycles to produce
  for (long i = 0; i < numCycles; i++) { // for the calculated length of time...
    digitalWrite(targetPin, HIGH); // write the buzzer pin high to push out the diaphram
    delayMicroseconds(delayValue); // wait for the calculated delay value
    digitalWrite(targetPin, LOW); // write the buzzer pin low to pull back the diaphram
    delayMicroseconds(delayValue); // wait again or the calculated delay value
  }
 digitalWrite(2, LOW);

}
```