Meow Box - Beta Preview 01

Updates: In an ideal situation, I am expecting my code to play the meow sound when people are far away from the box and when they approach they will get rick rolled.

Problem: The sensor is trying to measure the distance at the same time playing the buzz sound. In addition, the sensor prints 0s before it prints out the actual distance between the installation and the object.

Possible solution: What I am trying to do is breaking the distance measuring loop. Only at a certain distance, for example, distance ≤ 35, I will stop the distance measuring and play rick roll. The meow sound will gets to play if the distance ≥ 35.

November 22th update: One of the possible solution would be:

```
//set an array of number that records the distance points
int measure[] = {0, 1, 2};
int count = 0;
...
//check the distance and record it with a variable
//also set the flag to false to break the distance checking loop
if (distance ≥ 10 && flag){
    count = measure[0];
    flag = false;
    }else if (distance ≥ 20 && flag){
        count = measure[1];
    }else{
        count = measure[2];
}
flag = true;
//play the music under a certain conditions
if (count == 0){
```

```
    play(); //this is to play rick roll
    }else{
    meow2();
}
```

The alternative way is to add a for loop around the if statements. This way, as soon as the distance goes above 35 for example, the if is no longer checkin for the distance.

Existing problem: The 0 does not goes away

November 23th Update:

After a few times of investigation, I found the problem that is causing the display of 0s is the **int led = LED_BUILTIN.** Now I take out the code and seems the code is working in the TinkerCad. The next step would be playing rick roll at a certain distance and play the meows at another certain distances.

```
//initialization

//cat meow goes here
#define POT       A0   // for Trinket, use 1 for #2, 3 for #3, 2 for #4
//   for Uno/Leo/Mega A0 to A5
// define serial if using debug on Uno/Leo/Mega, Trinket/Gemma comment out
#define SERIAL

//rick roll goes here
#define  a3f    208     // 208 Hz
#define  b3f    233     // 233 Hz
#define  b3     247     // 247 Hz
#define  c4     261     // 261 Hz MIDDLE C
#define  c4s    277     // 277 Hz
#define  e4f    311     // 311 Hz
#define  f4     349     // 349 Hz
#define  a4f    415     // 415 Hz
#define  b4f    466     // 466 Hz
#define  b4     493     // 493 Hz
#define  c5     523     // 523 Hz
```

```
#define  c5s    554     // 554 Hz
#define  e5f    622     // 622 Hz
#define  f5     698     // 698 Hz
#define  f5s    740     // 740 Hz
#define  a5f    831     // 831 Hz

#define rest    -1

//int led = LED_BUILTIN;

volatile int beatlength = 100; // determines tempo
float beatseparationconstant = 0.3;

int threshold;

int a; // part index
int b; // song index
int c; // lyric index

// Parts 1 and 2 (Intro)

int song1_intro_melody[] =
{c5s, e5f, e5f, f5, a5f, f5s, f5, e5f, c5s, e5f, rest, a4f, a4f};

int song1_intro_rhythmn[] =
{6, 10, 6, 6, 1, 1, 1, 1, 6, 10, 4, 2, 10};

// Parts 3 or 5 (Verse 1)

int song1_verse1_melody[] =
{ rest, c4s, c4s, c4s, c4s, e4f, rest, c4, b3f, a3f,
rest, b3f, b3f, c4, c4s, a3f, a4f, a4f, e4f,
rest, b3f, b3f, c4, c4s, b3f, c4s, e4f, rest, c4, b3f, b3f, a3f,
rest, b3f, b3f, c4, c4s, a3f, a3f, e4f, e4f, e4f, f4, e4f,
c4s, e4f, f4, c4s, e4f, e4f, e4f, f4, e4f, a3f,
rest, b3f, c4, c4s, a3f, rest, e4f, f4, e4f
};

int song1_verse1_rhythmn[] =
{ 2, 1, 1, 1, 1, 2, 1, 1, 1, 5,
1, 1, 1, 1, 3, 1, 2, 1, 5,
1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 3,
1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 4,
```

```
5, 1, 1, 1, 1, 1, 1, 1, 2, 2,
2, 1, 1, 1, 3, 1, 1, 1, 3
};

const char* lyrics_verse1[] =
{ "We're ", "no ", "strangers ", "", "to ", "love ", "", "\r\n",
"You ", "know ", "the ", "rules ", "and ", "so ", "do ", "I\r\n",
"A ", "full ", "commitment's ", "", "", "what ", "I'm ", "thinking ", "", "of", "\r\n",
"You ", "wouldn't ", "", "get ", "this ", "from ", "any ", "", "other ", "", "guy\r\n",
"I ", "just ", "wanna ", "", "tell ", "you ", "how ", "I'm ", "feeling", "\r\n",
"Gotta ", "", "make ", "you ", "understand", "", "\r\n"
};

// Parts 4 or 6 (Chorus)

int song1_chorus_melody[] =
{ b4f, b4f, a4f, a4f,
f5, f5, e5f, b4f, b4f, a4f, a4f, e5f, e5f, c5s, c5, b4f,
c5s, c5s, c5s, c5s,
c5s, e5f, c5, b4f, a4f, a4f, a4f, e5f, c5s,
b4f, b4f, a4f, a4f,
f5, f5, e5f, b4f, b4f, a4f, a4f, a5f, c5, c5s, c5, b4f,
c5s, c5s, c5s, c5s,
c5s, e5f, c5, b4f, a4f, rest, a4f, e5f, c5s, rest
};

int song1_chorus_rhythmn[] =
{ 1, 1, 1, 1,
3, 3, 6, 1, 1, 1, 1, 3, 3, 3, 1, 2,
1, 1, 1, 1,
3, 3, 3, 1, 2, 2, 2, 4, 8,
1, 1, 1, 1,
3, 3, 6, 1, 1, 1, 1, 3, 3, 3, 1, 2,
1, 1, 1, 1,
3, 3, 3, 1, 2, 2, 2, 4, 8, 4
};

const char* lyrics_chorus[] =
{ "Never ", "", "gonna ", "", "give ", "you ", "up\r\n",
```

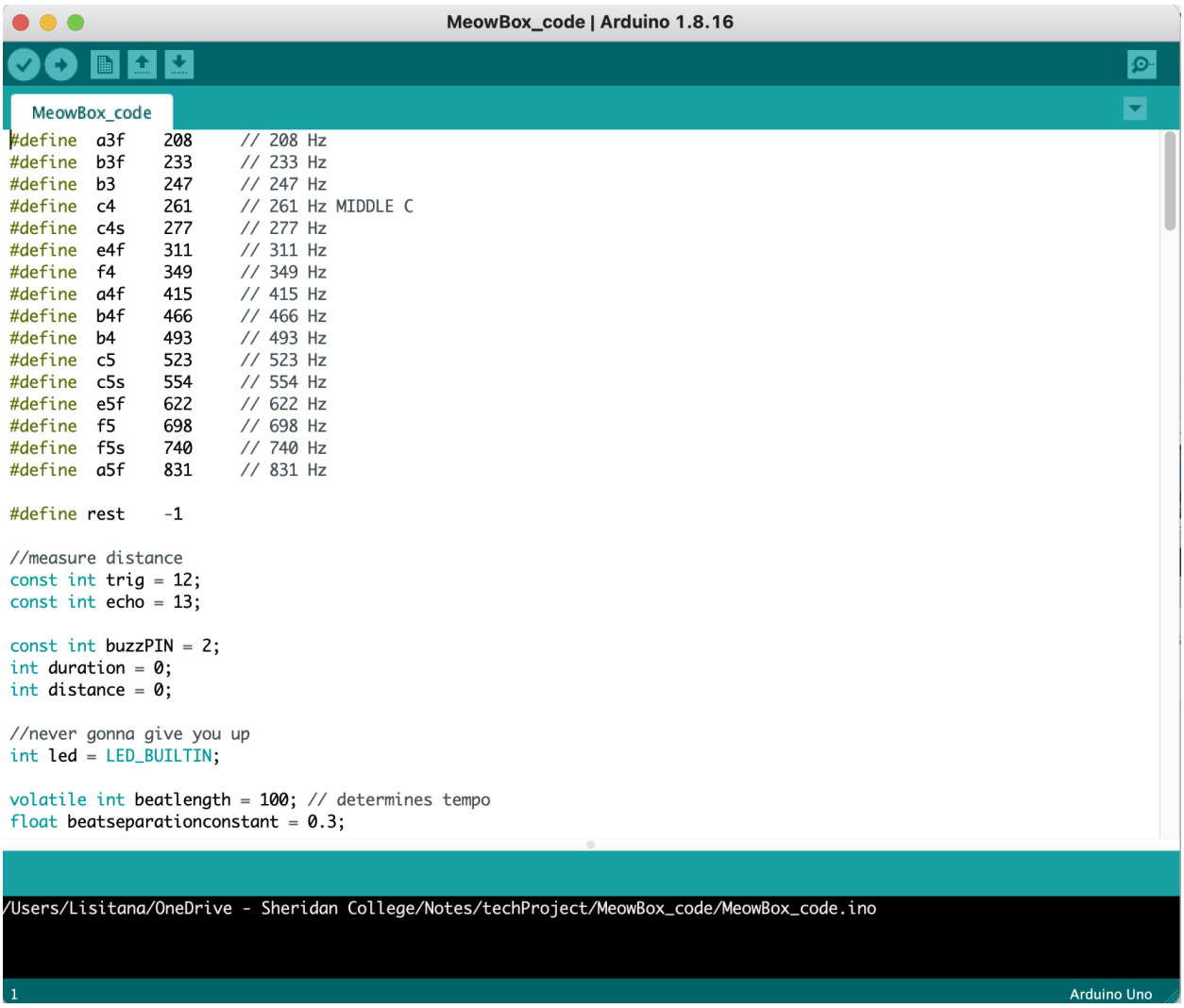
```
"Never ", "", "gonna ", "", "let ", "you ", "down", "", "\r\n",
"Never ", "", "gonna ", "", "run ", "around ", "", "", "", "and ", "desert ", "", "you\r\n",
"Never ", "", "gonna ", "", "make ", "you ", "cry\r\n",
"Never ", "", "gonna ", "", "say ", "goodbye ", "", "", "\r\n",
"Never ", "", "gonna ", "", "tell ", "a ", "lie ", "", "", "and ", "hurt ", "you\r\n"
};

//distance measuring
const int trig = 12; //output
const int echo = 13; //input

const int buzzPIN = 2;

int duration = 0;
int distance = 0;

void setup()
{
pinMode(trig , OUTPUT);
pinMode(echo , INPUT);

pinMode(buzzPIN, OUTPUT);

//rick roll goes here
//pinMode(led, OUTPUT);
//digitalWrite(led, LOW);

a = 4;
b = 0;
c = 0;

Serial.begin(9600);
}//end of void setup

void loop()
{
//loop the distance check

//checking the condition
for (int i = 0; i < 1; i++){
digitalWrite(trig , HIGH);
digitalWrite(trig , LOW);
```

```
duration = pulseIn(echo , HIGH);
distance = (duration/2) / 28.5 ;
Serial.println(distance);
}

if (distance <= 30 ){
play();
}else{
meow2();
delay(random(1000, 4000));
}

// if ( distance <= 50 )
// {
//
// }
// else
// {
//
// }

}//end of void loop

//meow2() function
void meow2() {  // cat meow (emphasis on "ow")
uint16_t i;
playTone(5100,55);       // "m" (short)
playTone(394,170);       // "eee" (long)
delay(300);              // wait a tiny bit
for(i=330; i<360; i+=2)  // vary "ooo" down
playTone(i,10);
playTone(5100,40);       // "w" (short)
}

//playTone() function
void playTone(uint16_t tone1, uint16_t duration) {
if(tone1 < 50 || tone1 > 15000) return;  // these do not play on a piezo
for (long i = 0; i < duration * 1000L; i += tone1 * 2) {
digitalWrite(buzzPIN, HIGH);
```

```
delayMicroseconds(tone1);
digitalWrite(buzzPIN, LOW);
delayMicroseconds(tone1);
}
}
//play() function
void play() {
int notelength;
if (a == 1 || a == 2) {
// intro
notelength = beatlength * song1_intro_rhythmn[b];
if (song1_intro_melody[b] > 0) {
//digitalWrite(led, HIGH);
tone(buzzPIN, song1_intro_melody[b], notelength);
}
b++;
if (b >= sizeof(song1_intro_melody) / sizeof(int)) {
a++;
b = 0;
c = 0;
}
}
else if (a == 3 || a == 5) {
// verse
notelength = beatlength * 2 * song1_verse1_rhythmn[b];
if (song1_verse1_melody[b] > 0) {
//digitalWrite(led, HIGH);
Serial.print(lyrics_verse1[c]);
tone(buzzPIN, song1_verse1_melody[b], notelength);
c++;
}
b++;
if (b >= sizeof(song1_verse1_melody) / sizeof(int)) {
a++;
b = 0;
```

```
c = 0;
}
}
else if (a == 4 || a == 6) {
// chorus
notelength = beatlength * song1_chorus_rhythmn[b];
if (song1_chorus_melody[b] > 0) {
//digitalWrite(led, HIGH);
Serial.print(lyrics_chorus[c]);
tone(buzzPIN, song1_chorus_melody[b], notelength);
c++;
}
b++;
if (b >= sizeof(song1_chorus_melody) / sizeof(int)) {
Serial.println("");
a++;
b = 0;
c = 0;
}
}
delay(notelength);
noTone(buzzPIN);
//digitalWrite(led, LOW);
delay(notelength * beatseparationconstant);
if (a == 7) { // loop back around to beginning of song
a = 1;
}
}
```

The code:


MeowBox_code | Arduino 1.8.16
MeowBox_code
#define a3f 208 // 208 Hz
#define b3f 233 // 233 Hz
#define b3 247 // 247 Hz
#define c4 261 // 261 Hz MIDDLE C
#define c4s 277 // 277 Hz
#define e4f 311 // 311 Hz
#define f4 349 // 349 Hz
#define a4f 415 // 415 Hz
#define b4f 466 // 466 Hz
#define b4 493 // 493 Hz
#define c5 523 // 523 Hz
#define c5s 554 // 554 Hz
#define e5f 622 // 622 Hz
#define f5 698 // 698 Hz
#define f5s 740 // 740 Hz
#define a5f 831 // 831 Hz
#define rest -1
//measure distance
const int trig = 12;
const int echo = 13;
const int buzzPIN = 2;
int duration = 0;
int distance = 0;
//never gonna give you up
int led = LED_BUILTIN;
volatile int beatlength = 100; // determines tempo
float beatseparationconstant = 0.3;
/Users/Lisitana/OneDrive - Sheridan College/Notes/techProject/MeowBox_code/MeowBox_code.ino
1
Arduino Uno

```
int threshold;

int a; // part index
int b; // song index
int c; // lyric index

boolean flag;

// Parts 1 and 2 (Intro)

int song1_intro_melody[] =
{c5s, e5f, e5f, f5, a5f, f5s, f5, e5f, c5s, e5f, rest, a4f, a4f};

int song1_intro_rhythmn[] =
{6, 10, 6, 6, 1, 1, 1, 1, 6, 10, 4, 2, 10};

// Parts 3 or 5 (Verse 1)

int song1_verse1_melody[] =
{ rest, c4s, c4s, c4s, c4s, e4f, rest, c4, b3f, a3f,
  rest, b3f, b3f, c4, c4s, a3f, a4f, a4f, e4f,
  rest, b3f, b3f, c4, c4s, b3f, c4s, e4f, rest, c4, b3f, b3f, a3f,
  rest, b3f, b3f, c4, c4s, a3f, a3f, e4f, e4f, e4f, f4, e4f,
  c4s, e4f, f4, c4s, e4f, e4f, e4f, f4, e4f, a3f,
  rest, b3f, c4, c4s, a3f, rest, e4f, f4, e4f
};



int a; // part index
int b; // song index
int c; // lyric index

boolean flag;

// Parts 1 and 2 (Intro)

int song1_intro_melody[] =
{c5s, e5f, e5f, f5, a5f, f5s, f5, e5f, c5s, e5f, rest, a4f, a4f};

int song1_intro_rhythmn[] =
{6, 10, 6, 6, 1, 1, 1, 1, 6, 10, 4, 2, 10};

// Parts 3 or 5 (Verse 1)

int song1_verse1_melody[] =
{ rest, c4s, c4s, c4s, c4s, e4f, rest, c4, b3f, a3f,
  rest, b3f, b3f, c4, c4s, a3f, a4f, a4f, e4f,
  rest, b3f, b3f, c4, c4s, b3f, c4s, e4f, rest, c4, b3f, b3f, a3f,
  rest, b3f, b3f, c4, c4s, a3f, a3f, e4f, e4f, e4f, f4, e4f,
  c4s, e4f, f4, c4s, e4f, e4f, e4f, f4, e4f, a3f,
  rest, b3f, c4, c4s, a3f, rest, e4f, f4, e4f
};
```

```
int song1_verse1_rhythmn[] =
{ 2, 1, 1, 1, 1, 2, 1, 1, 1, 5,
  1, 1, 1, 1, 3, 1, 2, 1, 5,
  1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 3,
  1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 4,
  5, 1, 1, 1, 1, 1, 1, 1, 2, 2,
  2, 1, 1, 1, 3, 1, 1, 1, 3
};

const char* lyrics_verse1[] =
{ "We're ", "no ", "strangers ", "", "to ", "love ", "", "\r\n",
  "You ", "know ", "the ", "rules ", "and ", "so ", "do ", "I\r\n",
  "A ", "full ", "commitment's ", "", "", "what ", "I'm ", "thinking ", "", "of", "\r\n",
  "You ", "wouldn't ", "", "get ", "this ", "from ", "any ", "", "other ", "", "guy\r\n",
  "I ", "just ", "wanna ", "", "tell ", "you ", "how ", "I'm ", "feeling", "\r\n",
  "Gotta ", "", "make ", "you ", "understand", "", "\r\n"
};

// Parts 4 or 6 (Chorus)

int song1_chorus_melody[] =
{ b4f, b4f, a4f, a4f,
  f5, f5, e5f, b4f, b4f, a4f, a4f, e5f, e5f, c5s, c5, b4f,
  c5s, c5s, c5s, c5s,
  c5s, e5f, c5, b4f, a4f, a4f, a4f, e5f, c5s,
  b4f, b4f, a4f, a4f,
  f5, f5, e5f, b4f, b4f, a4f, a4f, a5f, c5, c5s, c5, b4f,
  c5s, c5s, c5s, c5s,
  c5s, e5f, c5, b4f, a4f, rest, a4f, e5f, c5s, rest
};



int song1_chorus_rhythmn[] =
{ 1, 1, 1, 1,
  3, 3, 6, 1, 1, 1, 1, 3, 3, 3, 1, 2,
  1, 1, 1, 1,
  3, 3, 3, 1, 2, 2, 2, 4, 8,
  1, 1, 1, 1,
  3, 3, 6, 1, 1, 1, 1, 3, 3, 3, 1, 2,
  1, 1, 1, 1,
  3, 3, 3, 1, 2, 2, 2, 4, 8, 4
};

const char* lyrics_chorus[] =
{ "Never ", "", "gonna ", "", "give ", "you ", "up\r\n",
  "Never ", "", "gonna ", "", "let ", "you ", "down", "", "\r\n",
  "Never ", "", "gonna ", "", "run ", "around ", "", "", "", "and ", "desert ", "", "you\r\n",
  "Never ", "", "gonna ", "", "make ", "you ", "cry\r\n",
  "Never ", "", "gonna ", "", "say ", "goodbye ", "", "", "\r\n",
  "Never ", "", "gonna ", "", "tell ", "a ", "lie ", "", "", "and ", "hurt ", "you\r\n"
};

void setup()
{
  pinMode(trig , OUTPUT);
  pinMode(echo , INPUT);

  pinMode(buzzPIN, OUTPUT);

  pinMode(led, OUTPUT);
  digitalWrite(led, LOW);
  Serial.begin(9600);
```

```
  flag = true;
  a = 4;
  b = 0;
  c = 0;

}
void loop()
{
  digitalWrite(trig , HIGH);
  delayMicroseconds(5);
  digitalWrite(trig , LOW);
  duration = pulseIn(echo , HIGH);
  distance = (duration / 2) / 28.5 ;
  Serial.println(distance);

  do {
    play();
  } while (distance > 20);

 //do {
 //   meow2();
  //  delay(random(1500, 5000));
  // } while (distance < 5);
}

void play() {
  int notelength;
  if (a == 1 || a == 2) {
    // intro
    notelength = beatlength * song1_intro_rhythmn[b];
    if (song1_intro_melody[b] > 0) {
      digitalWrite(led, HIGH);


      tone(buzzPIN, song1_intro_melody[b], notelength);
    }
    b++;
    if (b >= sizeof(song1_intro_melody) / sizeof(int)) {
      a++;
      b = 0;
      c = 0;
    }
  }
  else if (a == 3 || a == 5) {
    // verse
    notelength = beatlength * 2 * song1_verse1_rhythmn[b];
    if (song1_verse1_melody[b] > 0) {
      digitalWrite(led, HIGH);
      Serial.print(lyrics_verse1[c]);
      tone(buzzPIN, song1_verse1_melody[b], notelength);
      c++;
    }
    b++;
    if (b >= sizeof(song1_verse1_melody) / sizeof(int)) {
      a++;
      b = 0;
      c = 0;
    }
  }
  else if (a == 4 || a == 6) {
    // chorus
    notelength = beatlength * song1_chorus_rhythmn[b];
    if (song1_chorus_melody[b] > 0) {
      digitalWrite(led, HIGH);
      Serial.print(lyrics_chorus[c]);
      tone(buzzPIN, song1_chorus_melody[b], notelength);
```

```
      c++;
    }
    b++;
    if (b >= sizeof(song1_chorus_melody) / sizeof(int)) {
      Serial.println("");
      a++;
      b = 0;
      c = 0;
    }
  }
  delay(notelength);
  noTone(buzzPIN);
  digitalWrite(led, LOW);
  delay(notelength * beatseparationconstant);
  if (a == 7) { // loop back around to beginning of song
    a = 1;
  }
}

void meow2() {  // cat meow (emphasis on "ow")
  uint16_t i;
  playTone(5100, 55);      // "m" (short)
  playTone(394, 170);      // "eee" (long)
  delay(30);               // wait a tiny bit
  for (i = 330; i < 360; i += 2) // vary "ooo" down
    playTone(i, 10);
  playTone(5100, 40);      // "w" (short)
}

void playTone(uint16_t tone1, uint16_t duration) {
  if (tone1 < 50 || tone1 > 15000) return; // these do not play on a piezo
  for (long i = 0; i < duration * 1000L; i += tone1 * 2) {
```

```
    digitalWrite(buzzPIN, HIGH);
    delayMicroseconds(tone1);
    digitalWrite(buzzPIN, LOW);
    delayMicroseconds(tone1);
  }
}
```

Future Plan / Future Challenges: If the distance measuring can be fixed, what I plan to do is figuring out Processing and Arduino code for uploading images of live and dead cat.

Back up Plan: If the time does not permit, the alternative plan for our meow box is to add more buzz sounds that can mesmerize people so they have no idea what is inside the box.