Intro to Arduino:

Build a Digital Music Box

The goal of this class

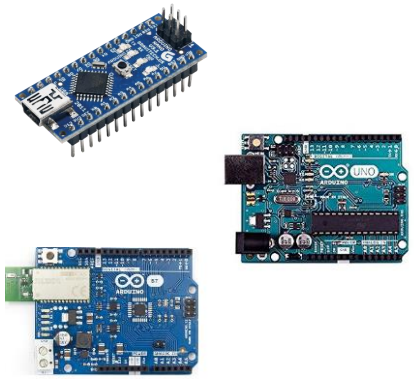
Answer #1: Build a digital music box

<u>The real answer:</u> To leave this class with an Arduino platform (the music box) and the skills needed to play around with it

What is an Arduino?

"Arduino is an open-source electronics platform based on easy-to-use ***hardware*** and ***software***" (arduino.cc)

Arduino boards



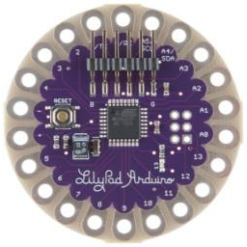
Arduino IDE (Integrated Development Environment)

"Integrated": contains all of the various software components needed to write code, compile it and upload it to an Arduino board

There is a wide variety of Arduino Boards

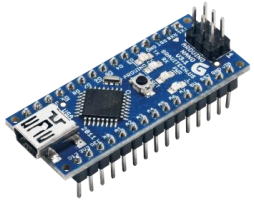
Uno

Mega



Lilypad

BT



Nano

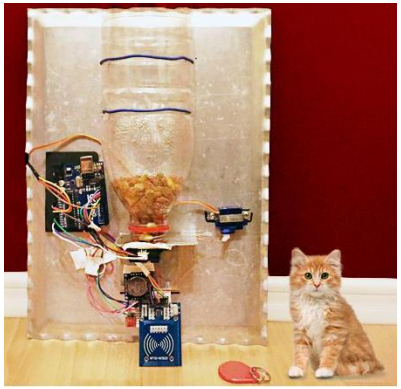
Pro Mini

There is also a wide variety of accessories such as “shields”

What does an Arduino do?

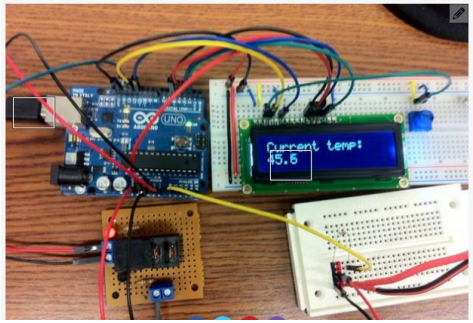
An Arduino is a general purpose tool that simply executes whatever instructions you give it

Pet Feeder



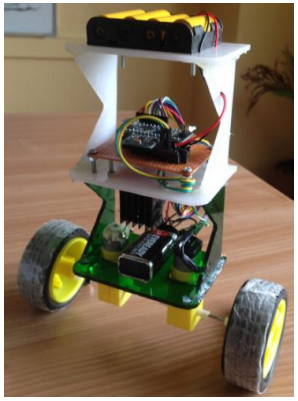
At 8am and 5pm send a signal to the motor so it rotates and dispenses food

Sous Vide Cooker



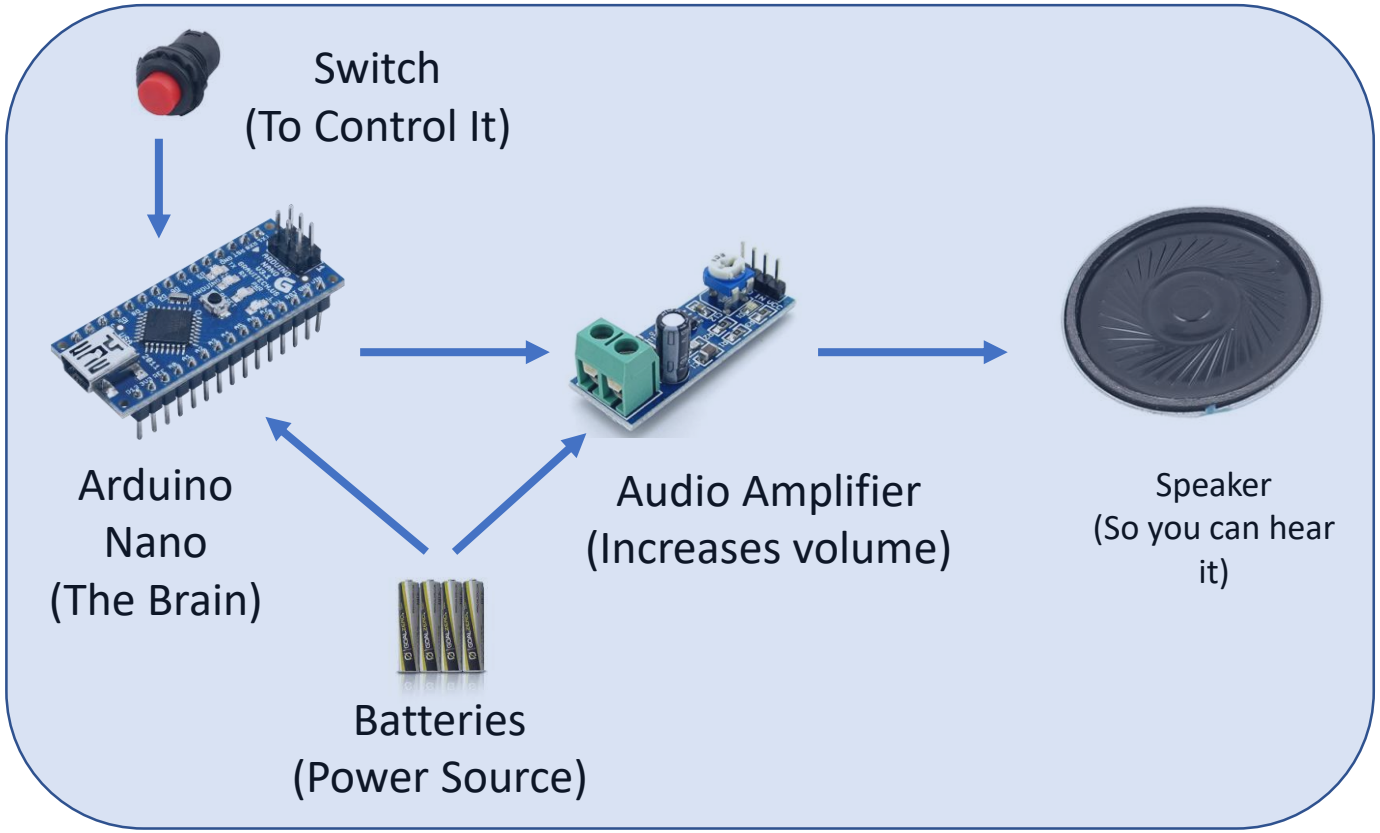
- If the temperature is too low, turn the heater on
- If the temperature is too high, turn the heater off

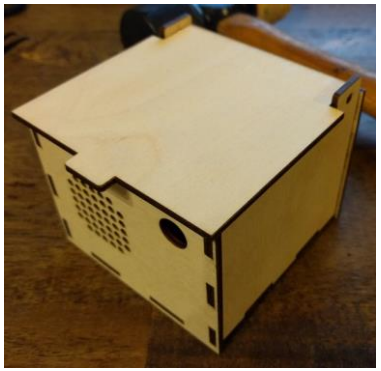
Self-Balancing Robot



- If leaning forward spin motors forward
- If leaning backwards, spin motors backwards

What are we building today?





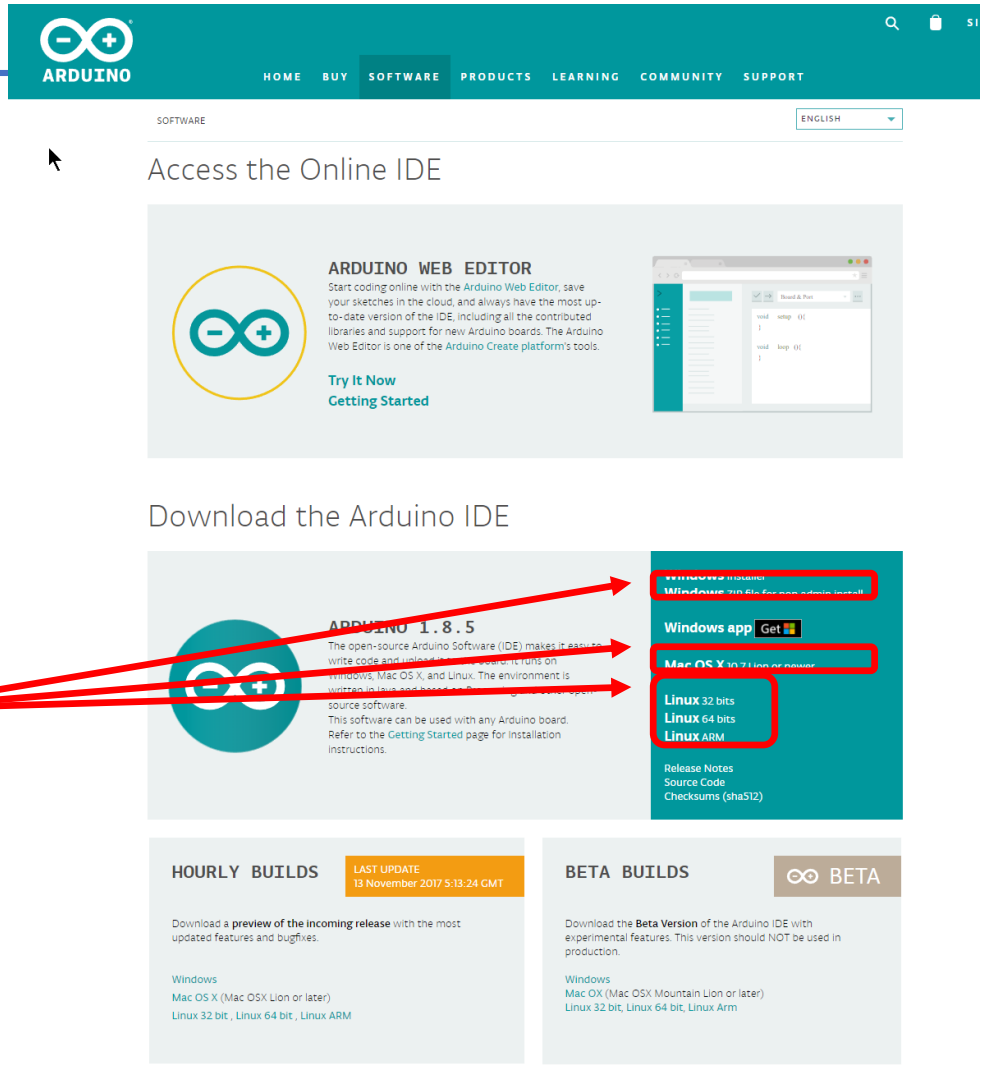
Best way to learn is hands-on so let’s get started

Download and install the Arduino IDE
- (and build the laser cut box)

Do a web search for “Arduino ide” and click on the appropriate link or just type it in:

https://www.arduino.cc/en/Main/Software

Download and install according to your platform



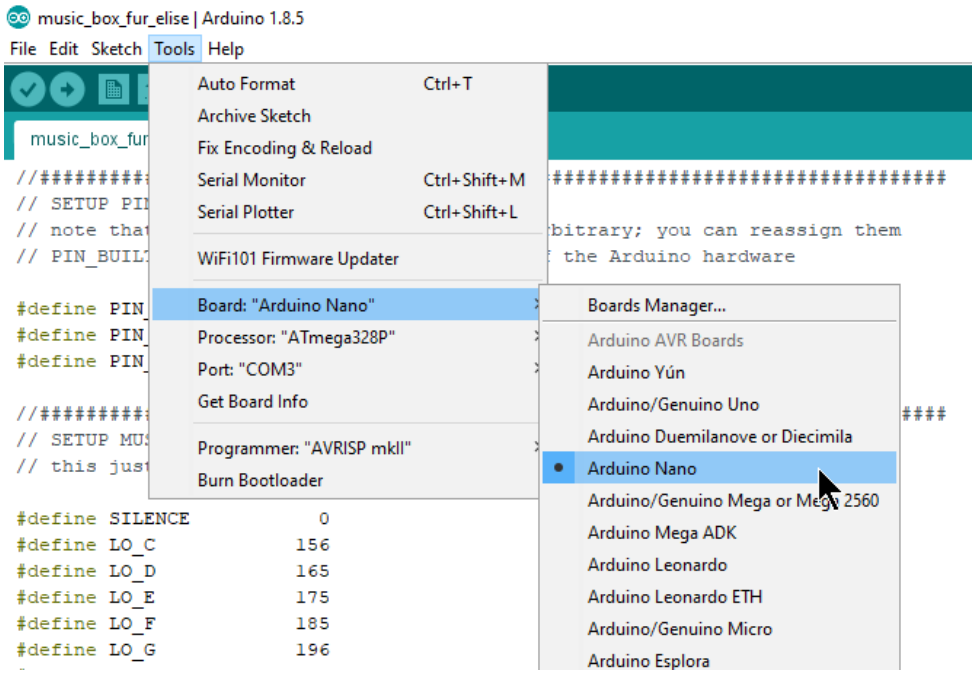
Configure the IDE – Select the correct board

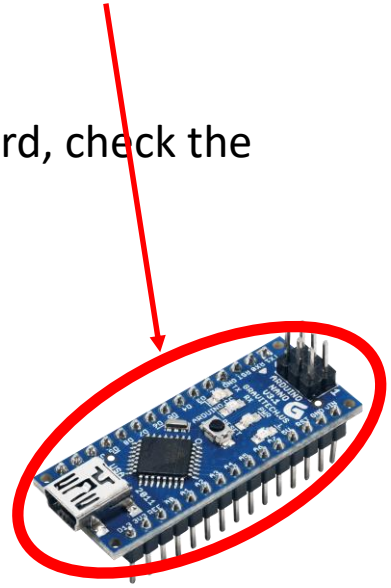
<u>Set the board type:</u>

Tools->Board->Arduino Nano

This refers to the Arduino board that is being used

When using a new board, check the documentation





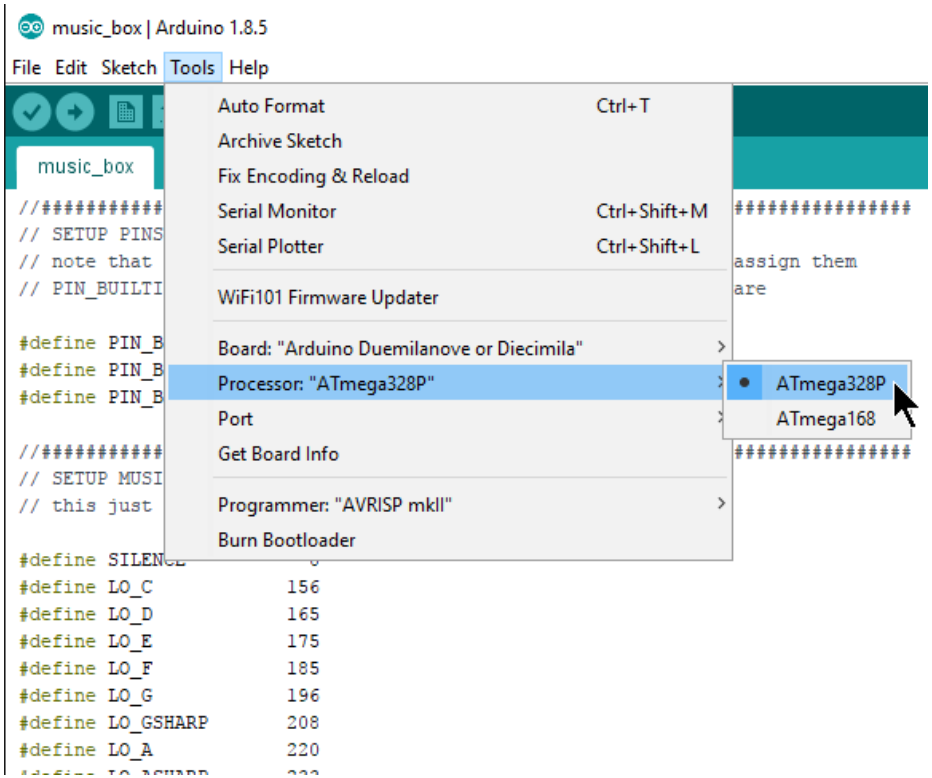
Configure the IDE – Select the correct processor

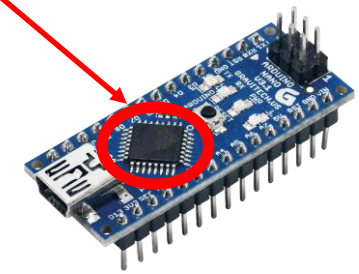
<u>Set the processor type:</u>

Tools->Processor->ATmega328P

This refers to the “CPU” or “microprocessor” (that is the primary hardware component) on the Arduino board.

Most boards seem to use the ATmega 328
When using a new board, check the documentation





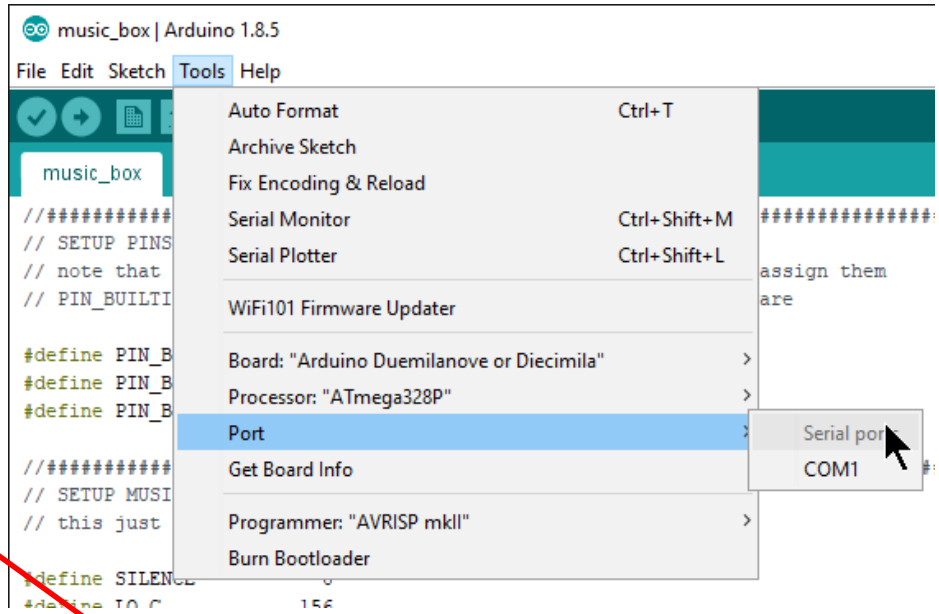
Configure the IDE – Select the correct port

Set the port:

Tools->Port->[this depends on your system]

This is the (typically USB) port on your computer that will be used to send data from the computer to the Arduino

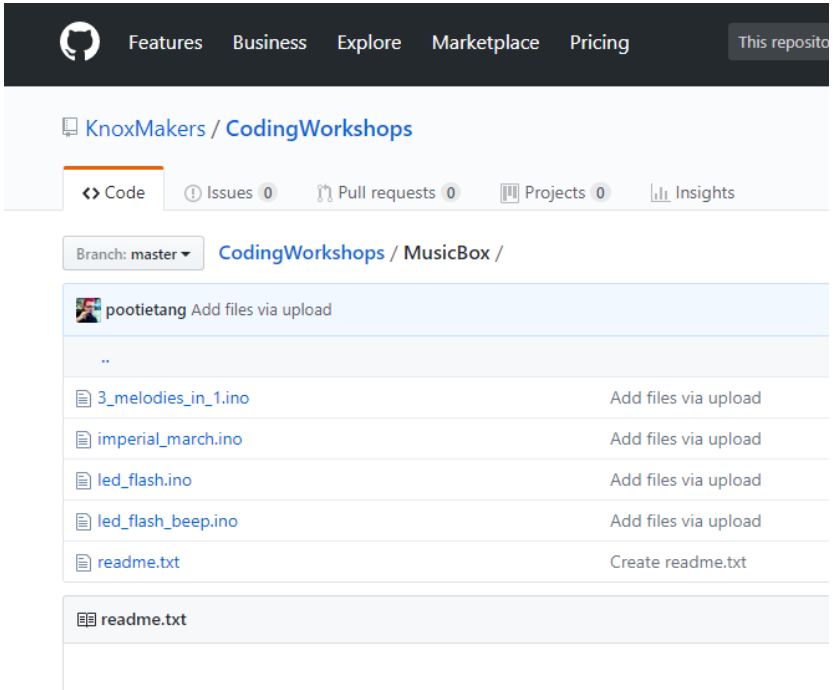
You often have to figure out which one is correct through trial and error



Download the led_flash sketch and upload to your Arduino

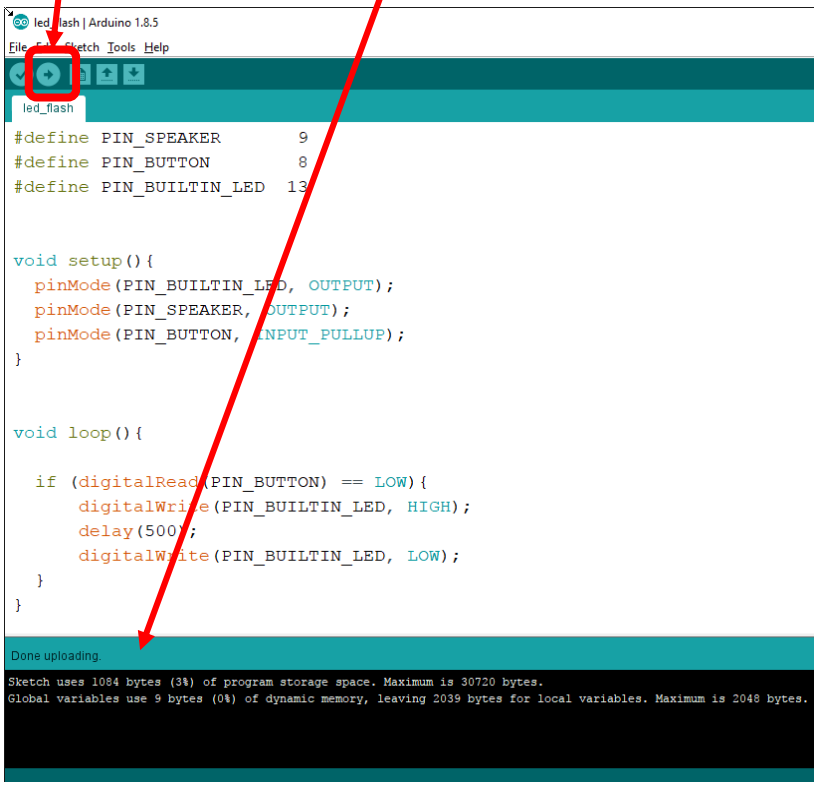
Download “led_flash.ino” from KM github

https://github.com/KnoxMakers/CodingWorkshops/tree/master/MusicBox



Upload to Arduino using the upload button:

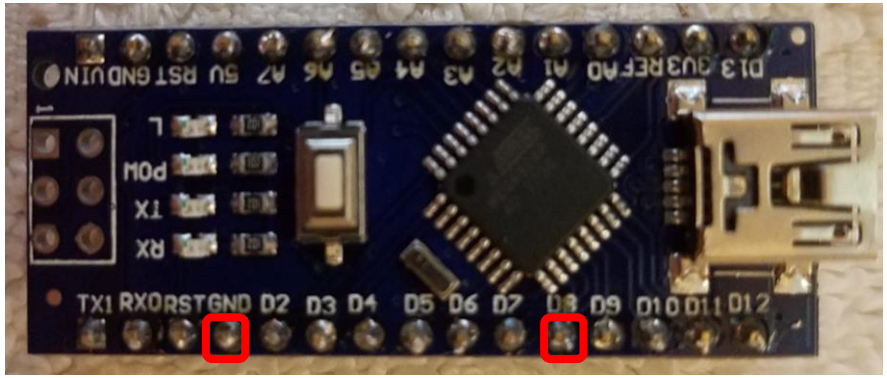
Look for the “Done uploading” (or errors) in the status bar



Connect the Pushbutton to your Arduino and press it!

One wire goes to GND, one wire goes to D8.
It doesn’t matter which wire goes to which pin
It doesn’t matter which pin labeled GND you use

When pressing the wires onto the pins use some force on the wire itself, not just the plastic housing, to help seat it.



The led_flash sketch explained

led_flash | Arduino 1.8.5

File Edit Sketch Tools Help

led_flash

```
#define PIN_SPEAKER        9
#define PIN_BUTTON         8
#define PIN_BUILTIN_LED   13
```

```
void setup(){
  pinMode(PIN_BUILTIN_LED, OUTPUT);
  pinMode(PIN_BUTTON, INPUT_PULLUP);
}
```

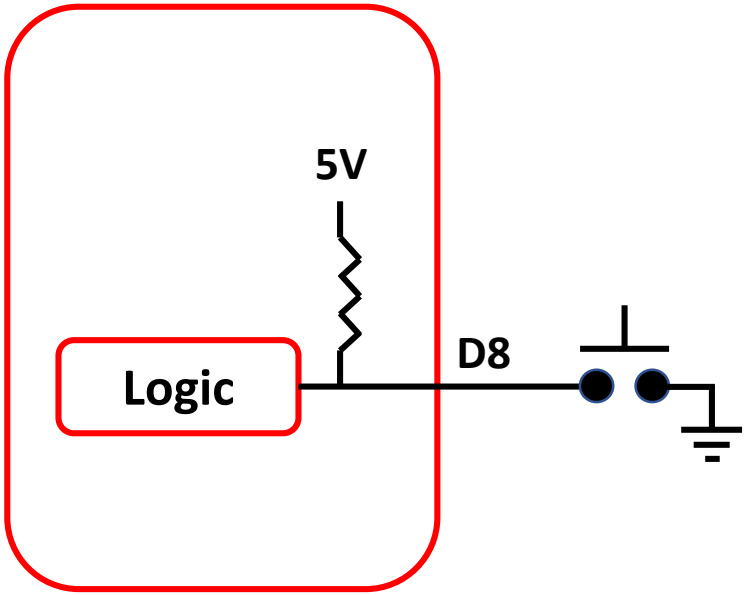
```
void loop(){

  if (digitalRead(PIN_BUTTON) == LOW){
      digitalWrite(PIN_BUILTIN_LED, HIGH);
      delay(500);
      digitalWrite(PIN_BUILTIN_LED, LOW);
  }
}
```

This sketch is broadly organized into 3 sections

- Definitions: which pins are used for what
  - This makes the code below easier to read
- Setup() – function is run one time, each time the Arduino is powered up
  - Used to set pins as inputs or outputs
- loop() – function is run over and over as long as the Arduino is powered up
  - Logic that determines what to do when button is pressed

INPUT_PULLUP explained

```
void setup(){
  pinMode(PIN_BUILTIN_LED, OUTPUT);
  pinMode(PIN_BUTTON, INPUT_PULLUP);
}
```



When Arduino pins are configured as inputs they can also be configured to have internal pull-up resistors enabled

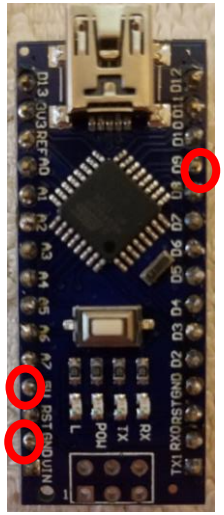
These resistors provide a gentle pull-up to the Vcc of the device

When the switch is pressed, it pulls the pin to ground much more strongly than the pull-up pulls to Vcc.

Arduinos also have a pull-down option

Let’s hook up the circuit so we have a speaker to use

1. Put AAA batteries in battery box
2. Turn battery box switch off
3. Connect battery box red leads to Arduino “5V” and Audio Amp “VCC”
4. Connect black leads to GND and GND



5. Connect the jumper wire from the Arduino D9 to the audio amp

6. Insert the speaker wire pins into the green terminals on the audio amplifier

Now modify the code to make it play a tone when you press the button

```
void setup(){
  pinMode(PIN_BUILTIN_LED, OUTPUT);
  pinMode(PIN_SPEAKER, OUTPUT);
  pinMode(PIN_BUTTON, INPUT_PULLUP);
}
```

- Add this line to the setup()

```
void loop(){

  if (digitalRead(PIN_BUTTON) == LOW){
      digitalWrite(PIN_BUILTIN_LED, HIGH);
      tone(PIN_SPEAKER,1000);
      delay(500);
      noTone(PIN_SPEAKER);
      digitalWrite(PIN_BUILTIN_LED, LOW);
  }
}
```

- Add these two lines to the code
- Upload the sketch
- Try it out!

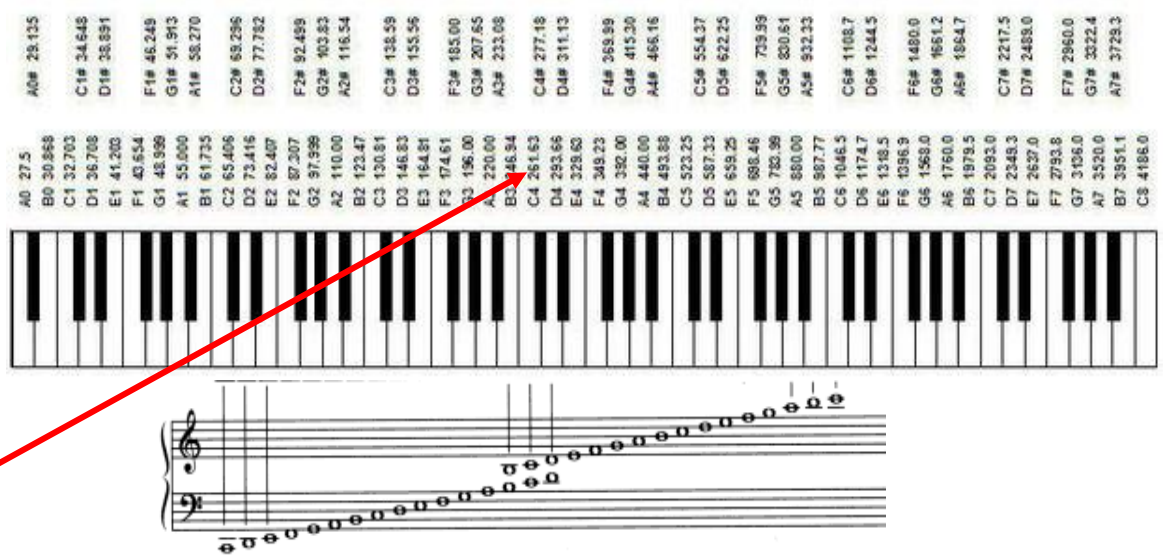
Now download fur_elise.ino, upload and try it out!

We added a few things to the definitions part of the sketch.

The first is a table that assigns names to frequencies, or notes.

In music, a key on the piano, or note on sheet music, corresponds to a particular frequency.

```
//#################################
// SETUP MUSICAL NOTES
// rather than remembering/typing the
// we associate musical note names wi
// the names in place of the numbers.

#define SILENCE          0
#define LO_C           156
#define LO_D           165
#define LO_E           175
#define LO_F           185
#define LO_G           196
#define LO_GSHARP      208
#define LO_A           220
#define LO_ASHARP      233
#define LO_B           246
#define MID_C          261
```



Now download fur_elise.ino, upload and try it out!

```
int note_count = 40;
int fur_elise[note_count][2] = {
  {HI_E,       150},
  {HI_DSHARP, 150},
  {HI_E,       150},
  {HI_DSHARP, 150},
  {HI_E,       150}
```

The melody is made up of 40 notes

We found with one version of the Arduino IDE we needed to replace “note_count” with “40” to avoid a compiler error

The melody is made up of pairs of note names and durations.

- Play the note HI_E for 150 milliseconds
- Then play the note HI_DSHARP for 150 milliseconds
- etc

The melody is stored in a data structure called a 2D matrix of int’s

int fur_elise[note_count] [2]

The matrix stores values that are of type “int” which means integers (-100, -4, 0, 2, 3940)

The name of the variable is “fur_elise”

The variable has 2 dimensions, rows and columns.
The number of rows is 40, the number of columns is 2

The Setup() portion is unchanged, just added comments

```
//##############################################################################
// This function runs once, when the Arduino first powers up.

void setup(){

  // PIN_BUILDIN_LED is the pin on the Arduino board that is connected to the LED o
  // We can change the status of this pin to turn the LED on and off
  pinMode(PIN_BUILTIN_LED, OUTPUT);

  // PIN_SPEAKER is the pin we will generate the music signal on
  pinMode(PIN_SPEAKER, OUTPUT);

  // PIN_BUTTON is the pin with the pushbutton connected ot it
  // It's an
  // PIN_BUTTON has is connected to ground via a pushbutton. INPUT_PULLUP mode
  // will pull the pin weakly HIGH. When the button is pressed, the pin will be
  // shorted to ground, pulling it strongly LOW.
  pinMode(PIN_BUTTON, INPUT_PULLUP);

}

//##############################################################################
```

The loop() portion is slightly different

```
void loop(){

  // PIN_BUTTON will only be LOW if the button is being pressed.
  if (digitalRead(PIN_BUTTON) == LOW){
      playFurElise();
  }
}
```

When the pushbutton is pressed, now we call the function playFurElise()

In software, a function is like a detour. When the Arduino gets to this point in the code it jumps to this function. When the function finishes, the Arduino continues from this point.

The playFurElise() portion is new, and a bit complicated

All this function does is go through the matrix of notes we looked at earlier, play them in sequence and blink the LED

Iterate through the matrix of notes - matrices start at 0 (not 1)

```
void playFurElise () {

  // variable "i" is our position within the MELODY structure
  for (int position=0; position<FUR_ELISE_NOTE_COUNT; position++){

    // if this NOTE isn't SILENCE, turn on the LED and emit a tone at FREQUENCY
    if (FUR_ELISE_MELODY[position][FREQUENCY] != SILENCE){
      digitalWrite(PIN_BUILTIN_LED, HIGH);
      tone(PIN_SPEAKER, FUR_ELISE_MELODY[position][FREQUENCY]);
    }

    // wait for the prescribed DURATION to elapse
    delay(FUR_ELISE_MELODY[position][DURATION]);

    // extinguish the LED and silence the buzzer
    digitalWrite(PIN_BUILTIN_LED, LOW);
    noTone(PIN_SPEAKER);

    // wait just a moment before moving to the next NOTE
    delay(50);
  }
}
```

Turn the LED on and off

tone() and noTone() play the notes

delay() lets the note play for the required duration

Finish it off

Put everything into the laser cut box

- Hot glue the speaker behind the grill
- Put the switch in the precut hole
- Put a small bit of Velcro on the battery box and the laser cut box to affix it in place
- Place everything else however you'd like
- The solder connections to the speaker are a weak point so affixing the audio amp with a little Velcro would be a good idea

Optional

- Download other sketches from the KM github
    - fur_elise.ino
    - Imperial_march.ino
    - 3_melodies_in_1.ino

What next?

- Have it play the melody twice
- Require 2 button presses to play it one time
- Write your own melody