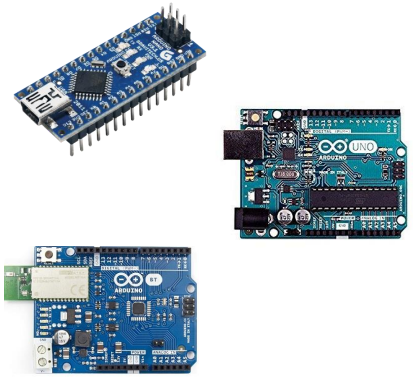
Intro to Arduino:

Build a Digital Music Box

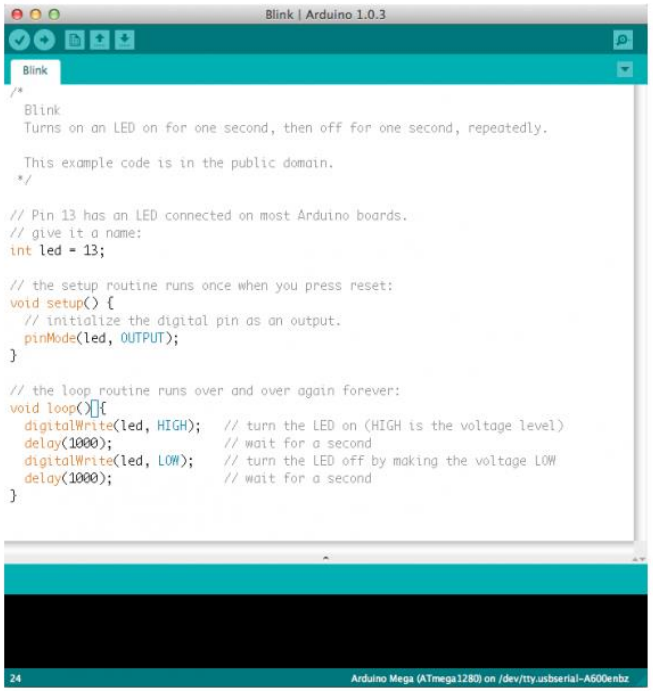
What is an Arduino?

Open Source
Hardware & Software

Arduino boards



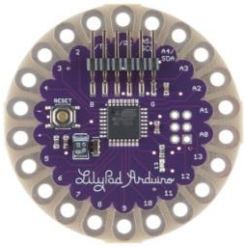
Arduino IDE and Libraries



There is a wide variety of Arduino Boards

Uno

Mega



Lilypad

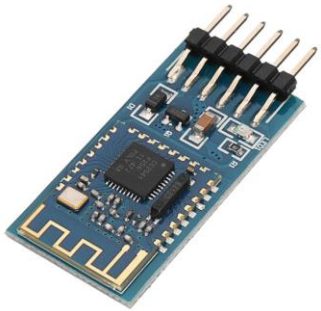
BT

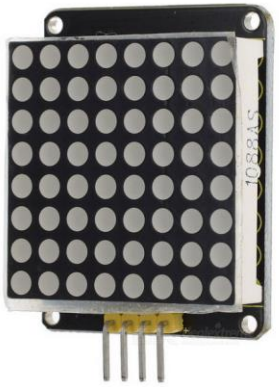
Nano

Pro Mini

Accessory “shields”

There is a wide range of free libraries

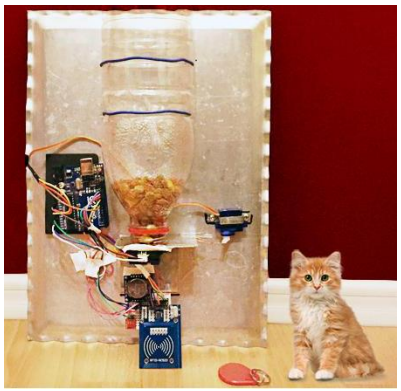




What can we do with an Arduino?

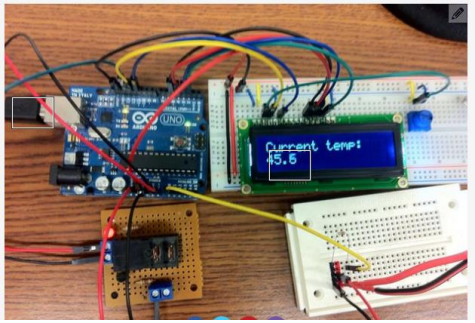
An Arduino is a general purpose tool that simply executes whatever instructions you give it

Pet Feeder



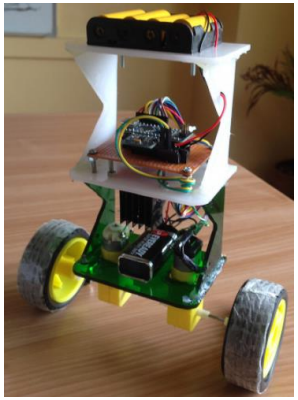
At 8am and 5pm send a signal to the motor so it rotates and dispenses food

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

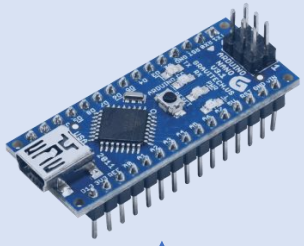


- If leaning forward spin motors forward
- If leaning backwards, spin motors backwards

What are we building today?

Switch
(To Control It)

Arduino
Nano
(The Brain)



Speaker

Batteries
(Power Source)

Start the IDE, Configure the Board and Processor

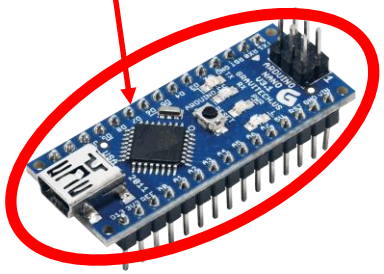
Set the board type:

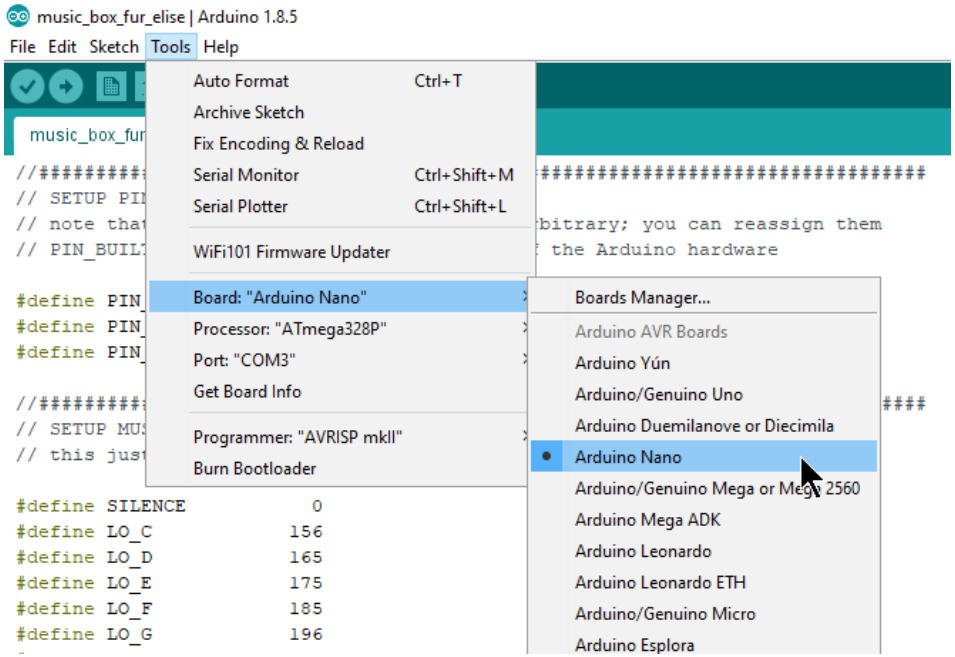
Tools->Board->Arduino Nano

Set the processor type:

Tools->Processor->ATmega328P (Old Bootloader)

This refers to the Arduino board that we are using





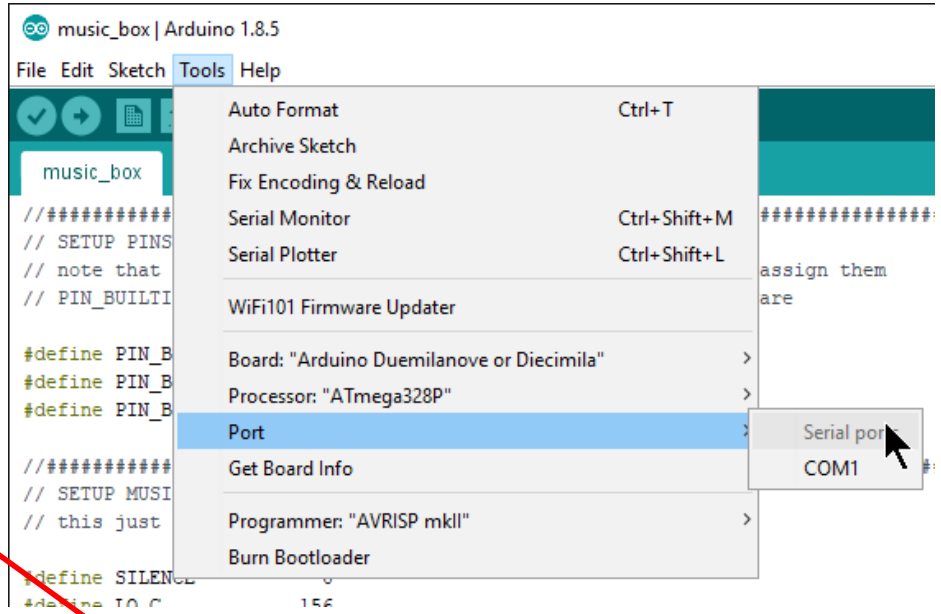
Configure the IDE – Select the correct port

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

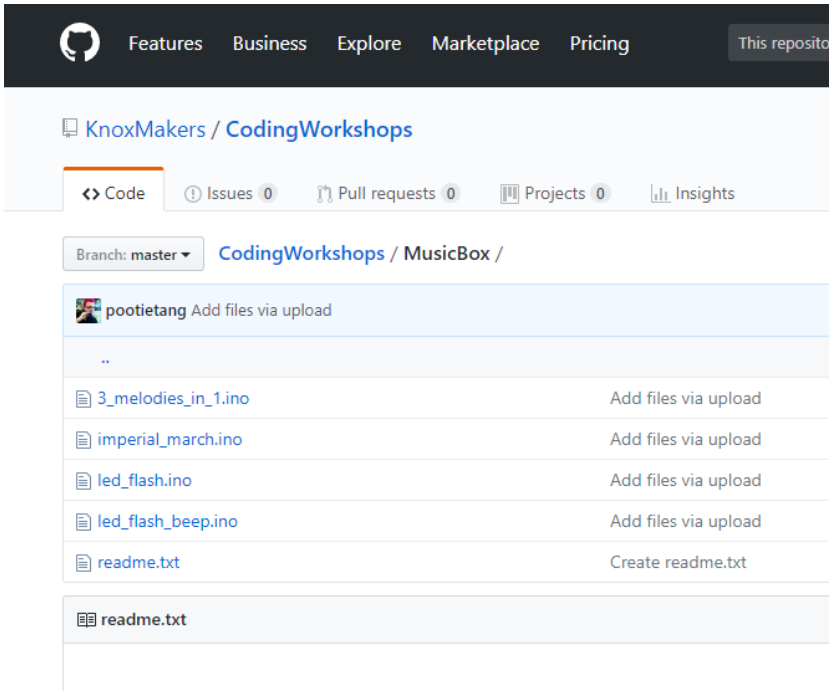


Download the led_flash sketch and upload to your Arduino

Download “led_flash.ino” from KM github

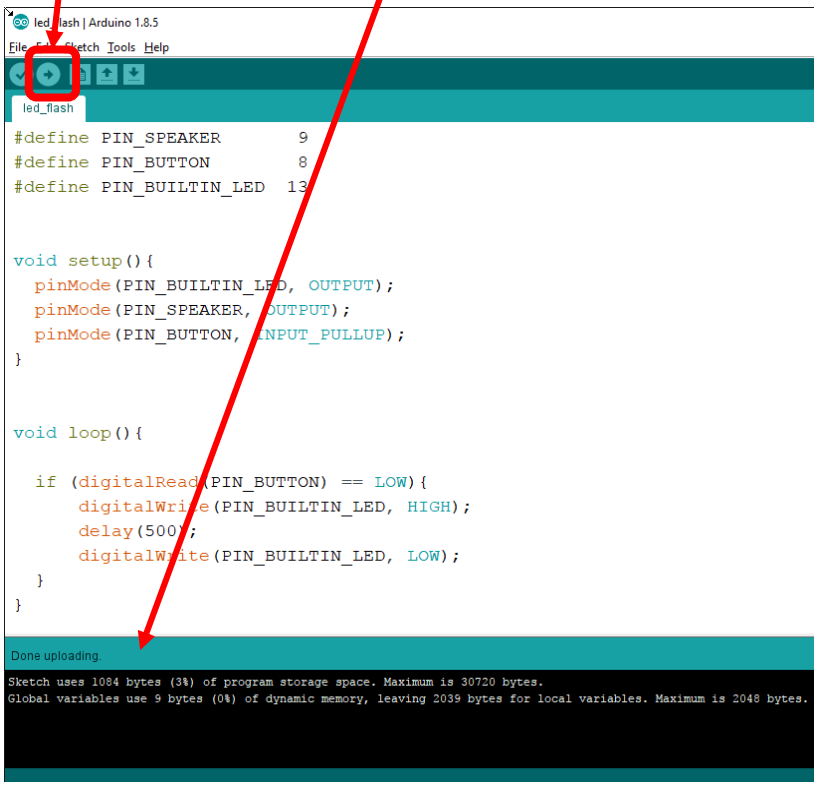
Web search “knox makers git hub”

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

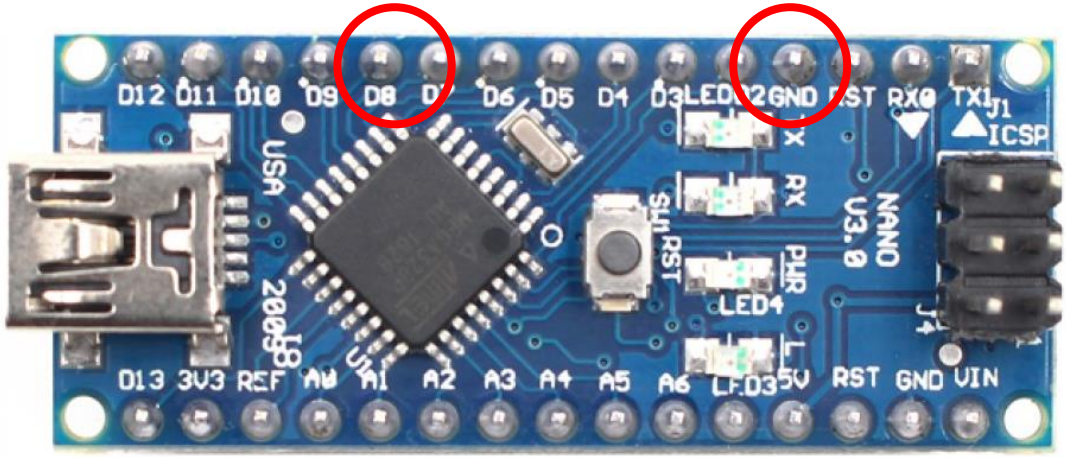


Connect the Pushbutton to your Arduino and press it!

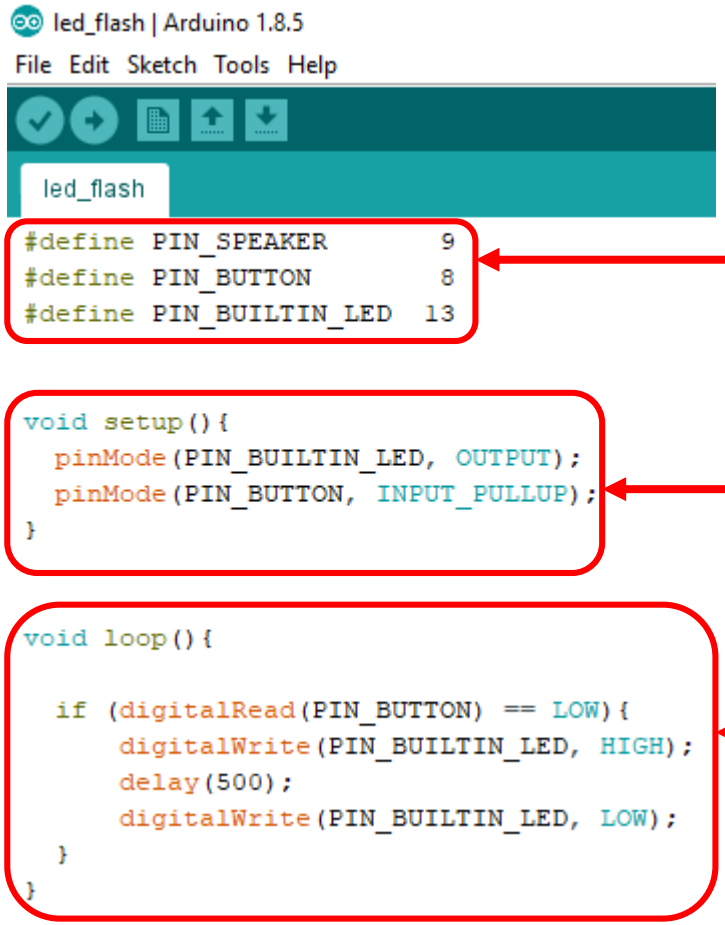
One wire goes to GND, one wire goes to D8

5th pin from left

4th pin from right



The led_flash sketch explained

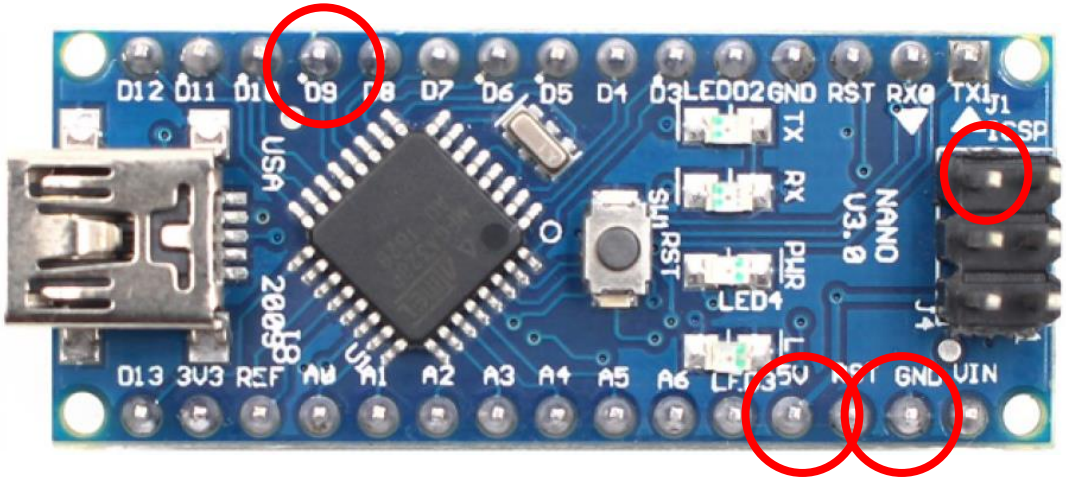


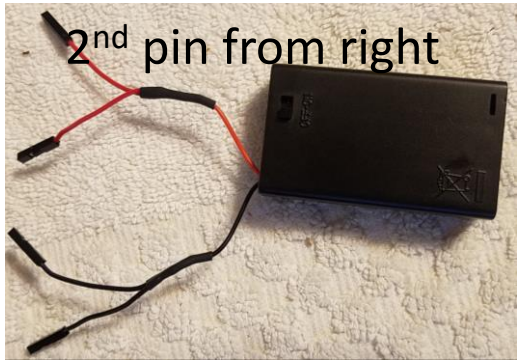
This sketch is broadly organized into 3 sections

- 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

Let’s hook up the rest of the circuit

one wire to GND
the other to D9





Red lead -> “5V”
Black lead -> ICSP GND

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() function

```
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 any of these and try them out

Web search “knox makers git hub”
https://github.com/KnoxMakers/CodingWorkshops/tree/master/MusicBox

jingle_bells.ino
fur_elise.ino
imperial_march.ino

Let’s look at jingle_bells.ino

```
void setup(){

  pinMode(PIN_BUILTIN_LED, OUTPUT);
  pinMode(PIN_SPEAKER, OUTPUT);
  pinMode(PIN_BUTTON, INPUT_PULLUP);
```

No change, defines our three pins

```
void loop(){

  if (digitalRead(PIN_BUTTON) == LOW){
        play_jingle_bells();
  }
}
```

Repeatedly checks if the button is pressed. If so, play jingle bells

Definition of Song Tempo and Note Duration

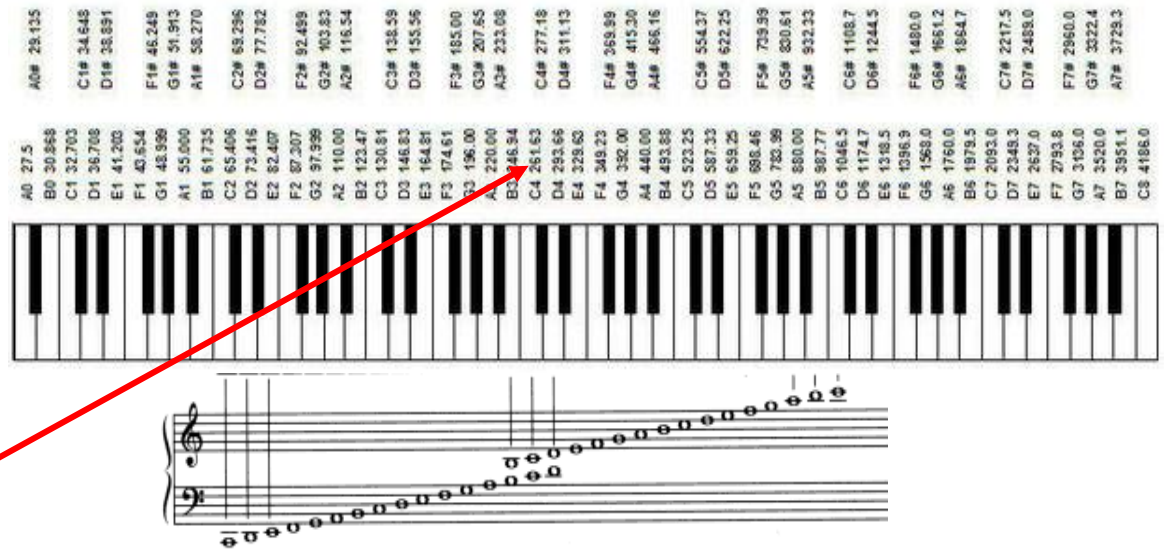
```
#define TEMPO 175


#define QUARTER 60000/TEMPO
#define HALF QUARTER*2
#define WHOLE HALF*2
#define EIGHTH QUARTER/2
#define SIXTEENTH EIGHTH/2
#define HALF_DOT HALF*3/2
#define QUARTER_DOT QUARTER*3/2
#define EIGHTH_DOT EIGHTH*3/2
#define SIXTEENTH_DOT SIXTEENTH*3/2
```

Table of musical notes

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



Definition of the song itself

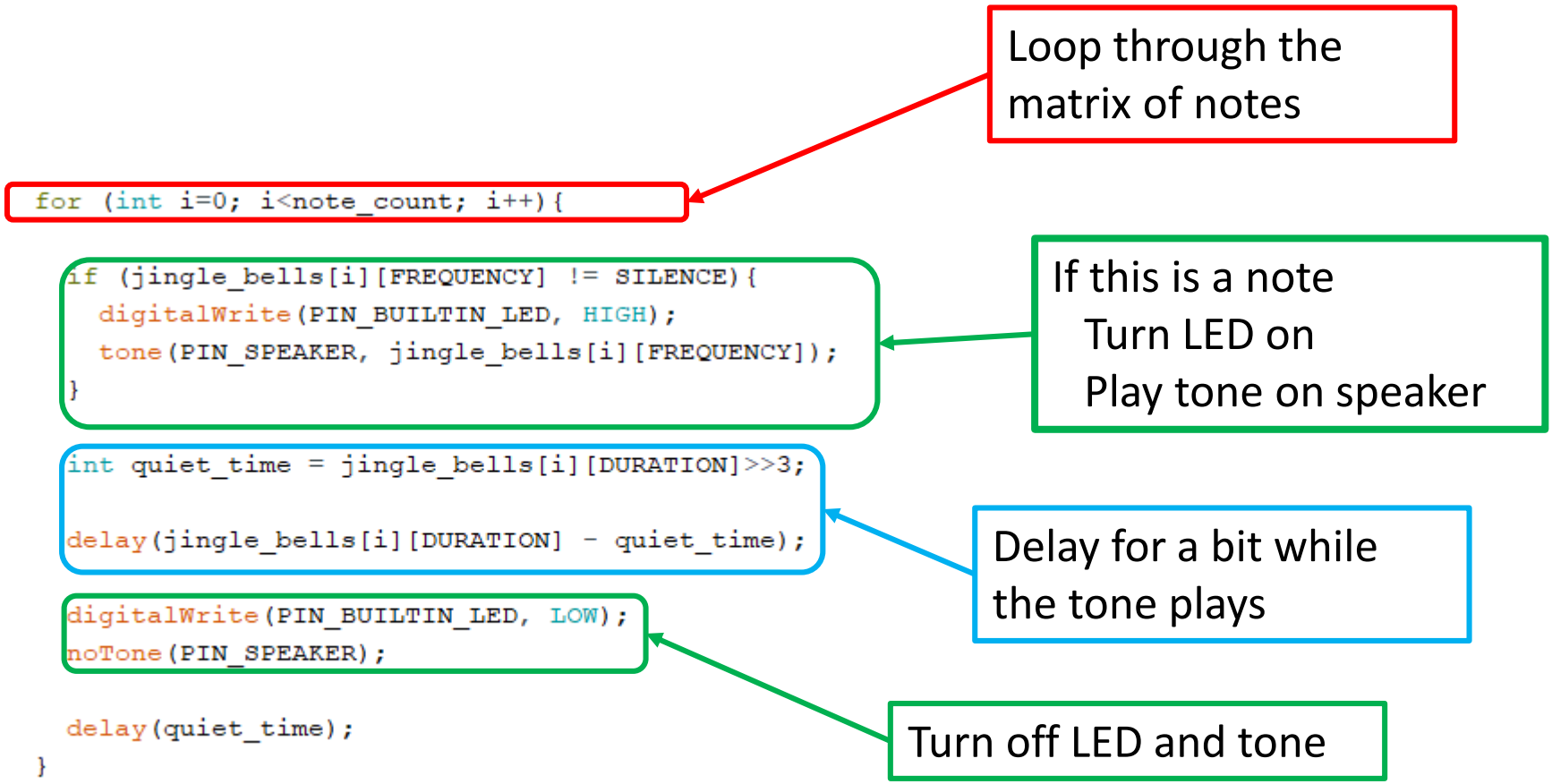
```
int note_count = 106;

int jingle_bells[note_count][2] = {
  {MID_C, QUARTER},
  {MID_A, QUARTER},
  {MID_G, QUARTER},
  {MID_F, QUARTER},
  {MID_C, HALF_DOT},
  {MID_C, EIGHTH},
  {MID_C, EIGHTH},
  {MID_C, QUARTER},
  {MID_A, QUARTER},
  {MID_G, QUARTER},
  {MID_F, QUARTER},
  {MID_D, HALF_DOT},
  {SILENCE, QUARTER},
  {MID_D, QUARTER},
  {MID_BFLAT, QUARTER},
  {MID_A, QUARTER},
  {MID_G, QUARTER},
```

Each row consists of one note (frequency and duration)

Further reading:
arrays and matrices in C

The playJingleBells() loop – play the notes



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

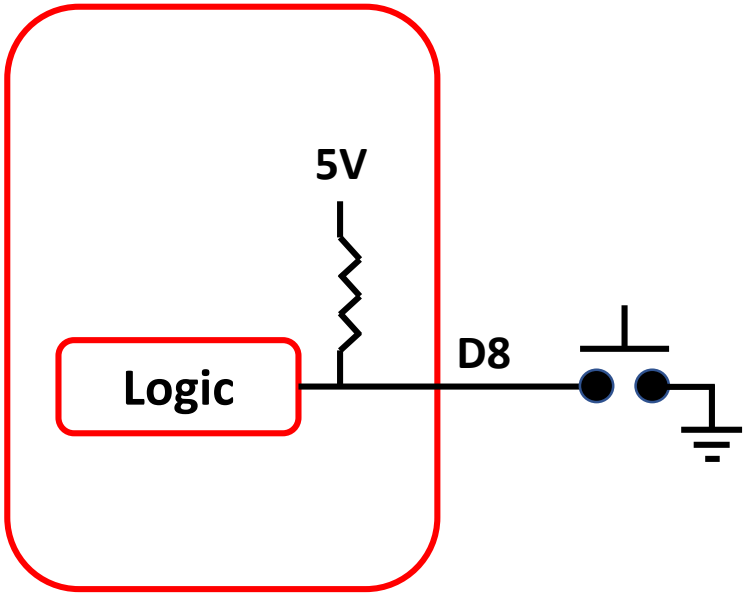
What next?

- Have it play the melody twice
- Require 2 button presses to play it one time
- Change which pins are used on the PC board
- Write your own melody

Appendix

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

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

```
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);
  }
}
```

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

Turn the LED on and off

tone() and noTone() play the notes

delay() lets the note play for the required duration