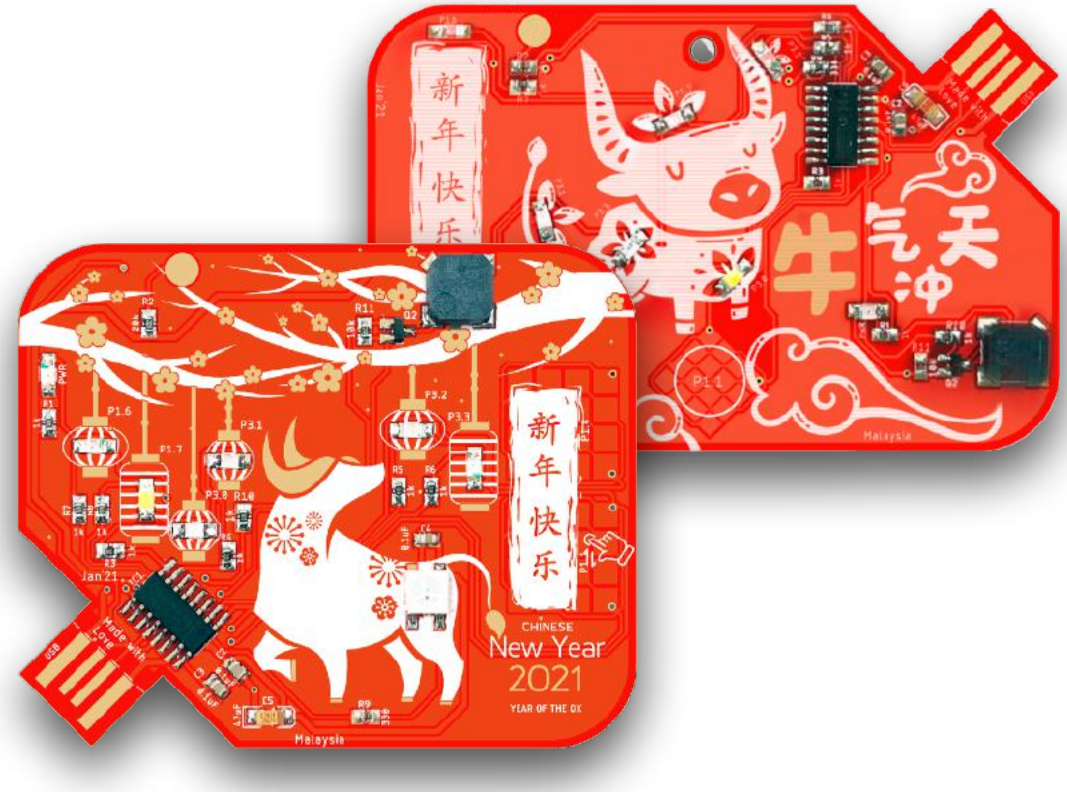
TomatoCube
Zoom Edition

Getting started on Programming with your Happy CNY Card



Yes, we are programmable!!

(Coding overview)

https://tomatocube.com/url/KjXMvP

Maintained By: Percy Chen

Last Updated: 02/2021

Things to learn today

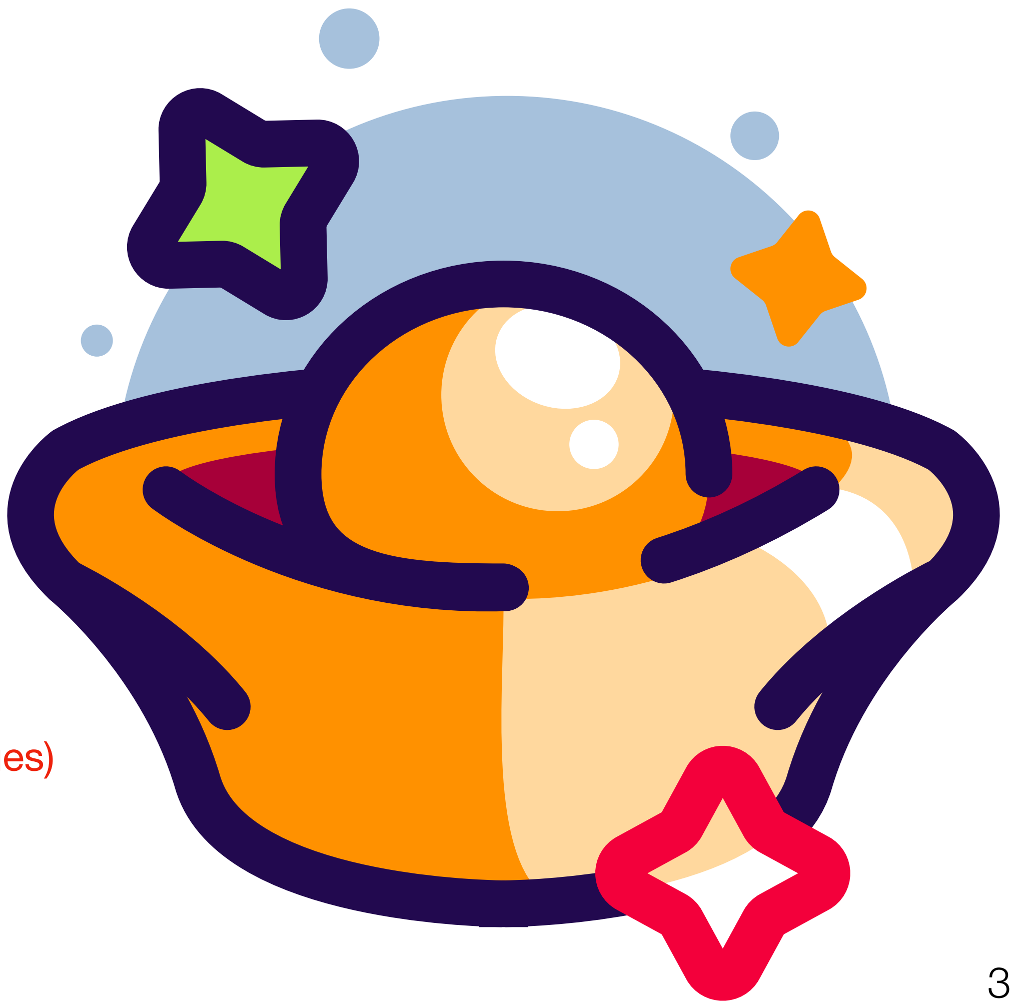
- Coding is one of the key skill-sets needed in the modern tomorrow world.
    - *It's not scary & difficult to get started in Coding. Furthermore, we have proof that it is not expensive as well.*

- **Target: (A) Those Curious about what is coding & its possibilities.**
  (B) Experience Arduino user, trying to play with an alternative Hardware.
  (C) This is not a comprehensive Beginner Coding class!!!

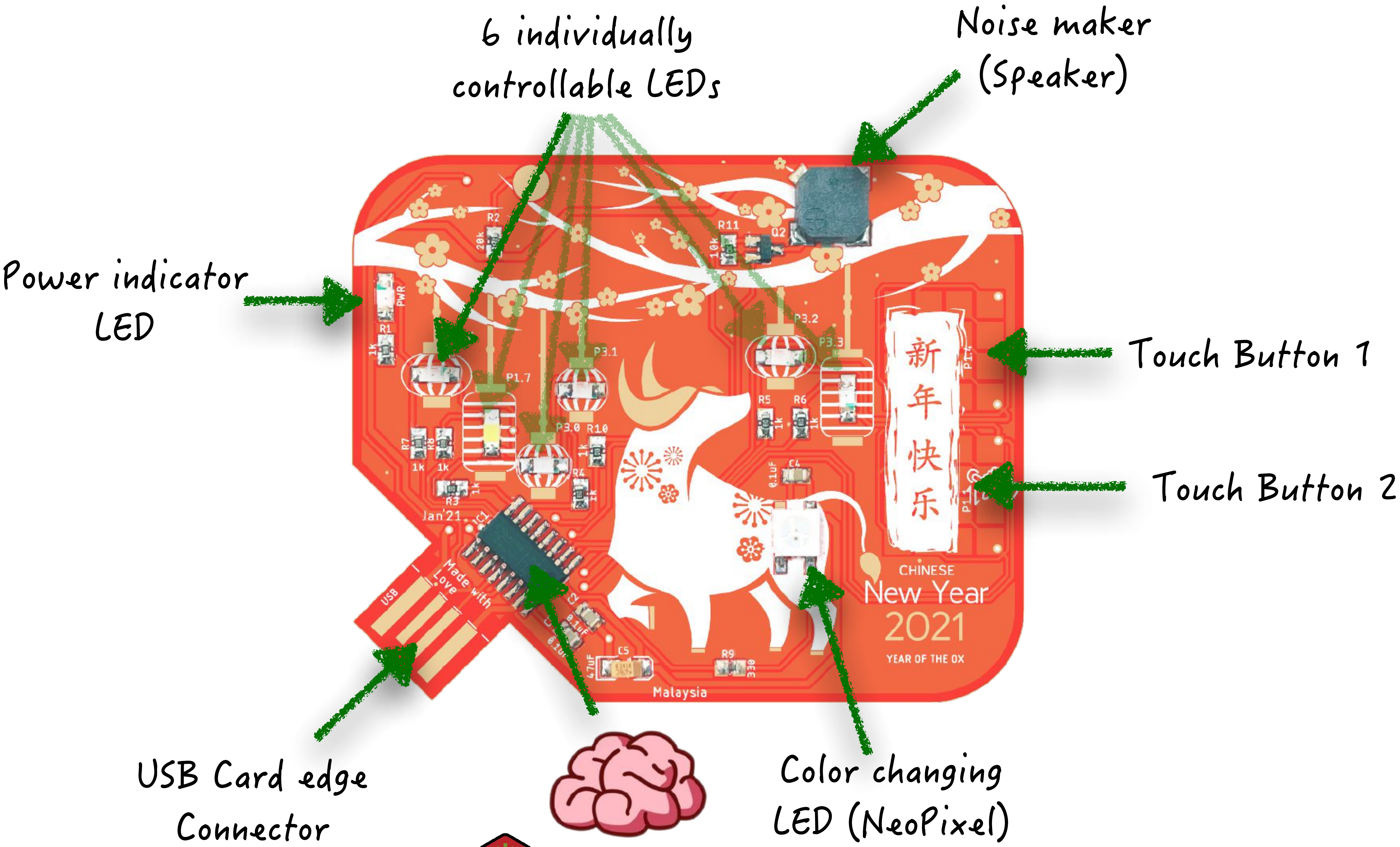
1 - 2 Hours.

Content

- **Getting Started:**
    - Happy CNY cards functions overview.
    - Arduino® software installation.
    - Microsoft® Windows driver, setup guide.
- **Coding:**
    - Code 1 - Blinking Light (Digital Out)
    - Code 2 - Blinking Light revisited (Helper API & Binary Bit Shift)
    - Code 3 - Mixing LED color (NeoPixel RGB)
    - Code 4 - Touch inputs (Capacitive touch sensing & conditional)
    - Code 5 - Learn about “Variables” (Variables & C Built-in functions)
    - Code 6 - Making Sound (Generating sound frequency)
    - Code 7 - Repetitive process (Loops & subroutine)
    - Code 8 - Code tidying & better Music code (Using Loops)
    - Code 9 - Mixing it together (Combining all the previous code & techniques)
    - Code X - Restoring the Default demo code.



Happy CNY Card (Design A) overview



6 individually controllable LEDs

Noise maker (Speaker)

Power indicator LED

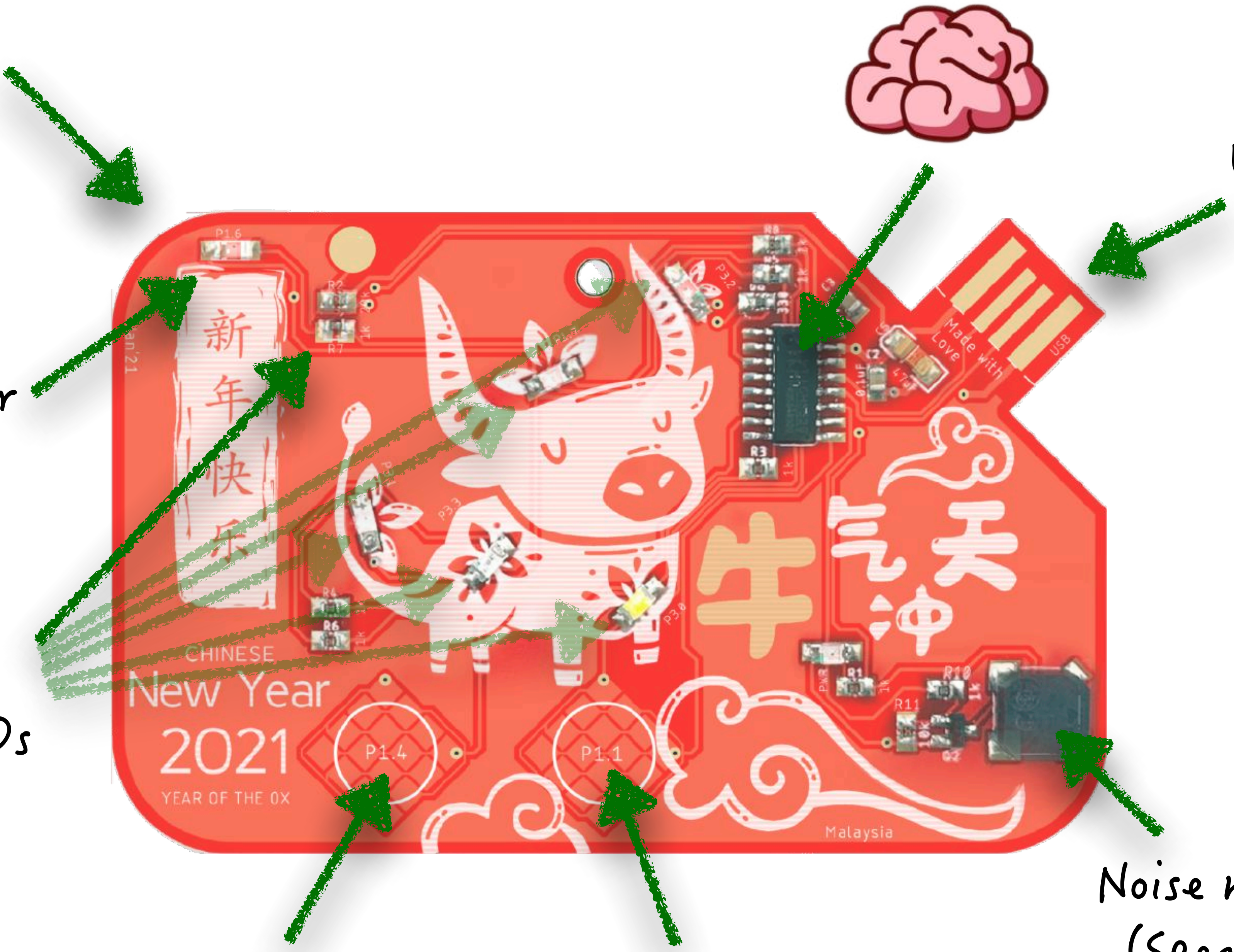
Touch Button 1

Touch Button 2

USB Card edge Connector

Color changing LED (NeoPixel)

Happy CNY Card (Design B) overview



Color changing LED (NeoPixel)

On the other-side

Power indicator LED

6 individually controllable LEDs

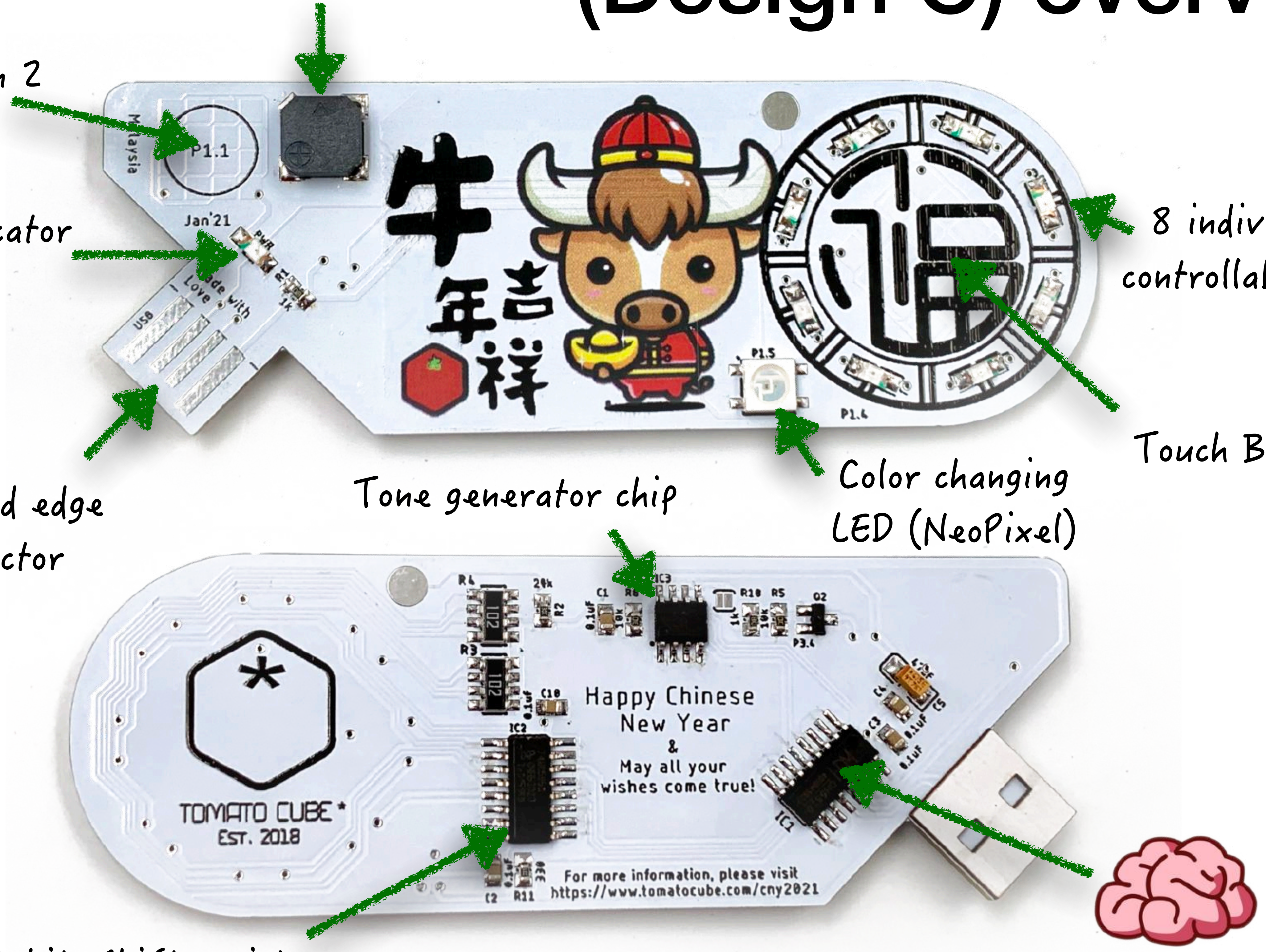
USB Card edge Connector

Noise maker (Speaker)

Touch Button 1

Touch Button 2

Happy CNY Card (Design C) overview



Noise maker (Speaker)

Touch Button 2

Power indicator LED

8 individually controllable LEDs

USB Card edge Connector

Touch Button 1

Tone generator chip

Color changing LED (NeoPixel)

8-bits Shift register

Where can I buy one?

Shopee

Shopee (TomatoCube*)

https://tomatocube.com/url/nevnV

Cytron Marketplace

Malaysia Largest Digital Maker Marketplace

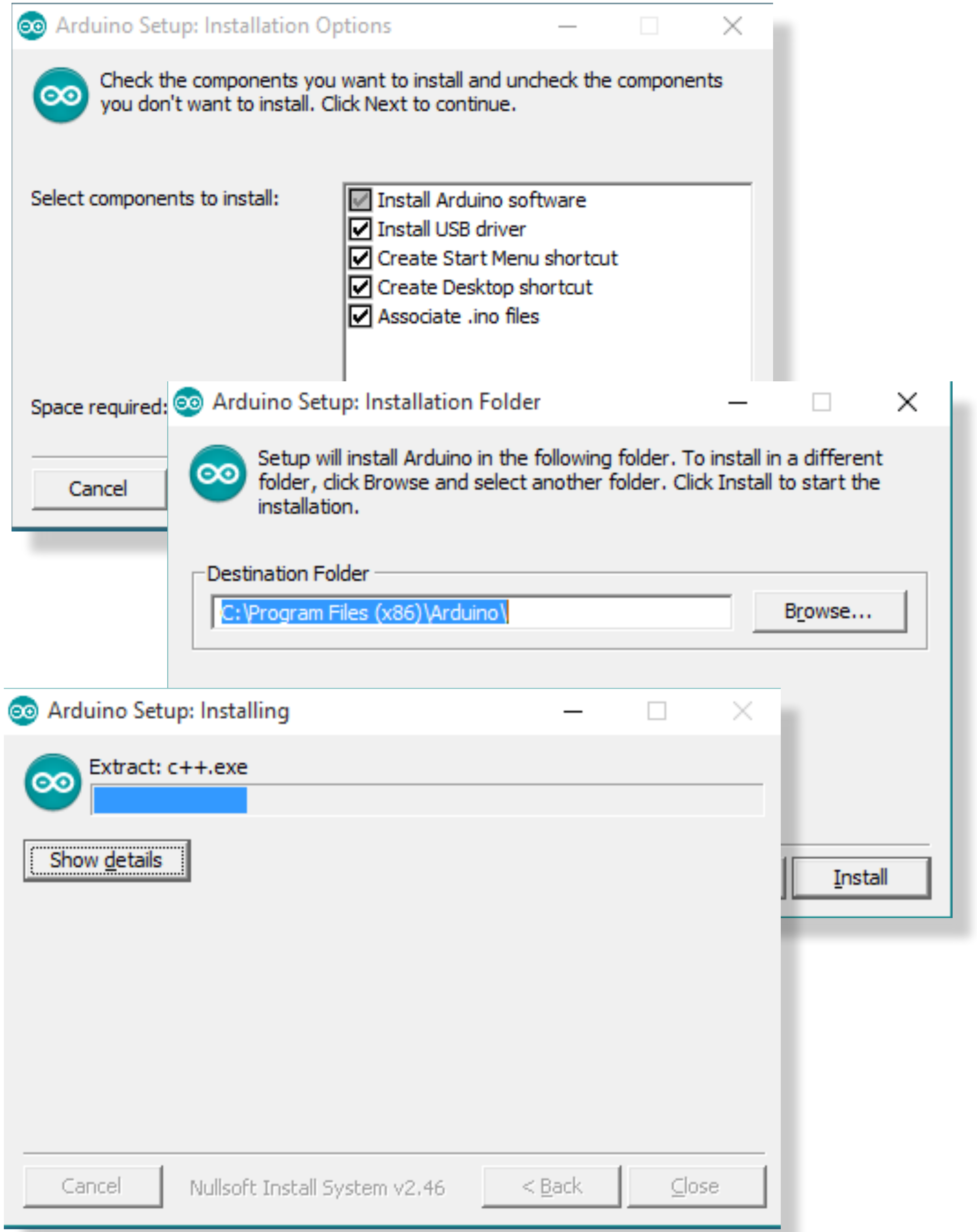
Cytron Marketplace

https://my.cytron.io/p-happy-cny-2021-programmable-pcb-card

Arduino® Desktop IDE

First you will need to install the Arduino® Desktop IDE.

Install the latest version of the Arduino Software (IDE) for the platform of your choice.
The software could be downloaded from
https://www.arduino.cc/en/Main/Software



Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

☑ Install Arduino software
☑ Install USB driver
☑ Create Start Menu shortcut
☑ Create Desktop shortcut
☑ Associate .ino files

Space required:

Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino\

Arduino Setup: Installing

Extract: c++.exe

Adding additional board support

- A fresh installation of the Arduino® IDE software will not be compatible with the Happy CNY Card.

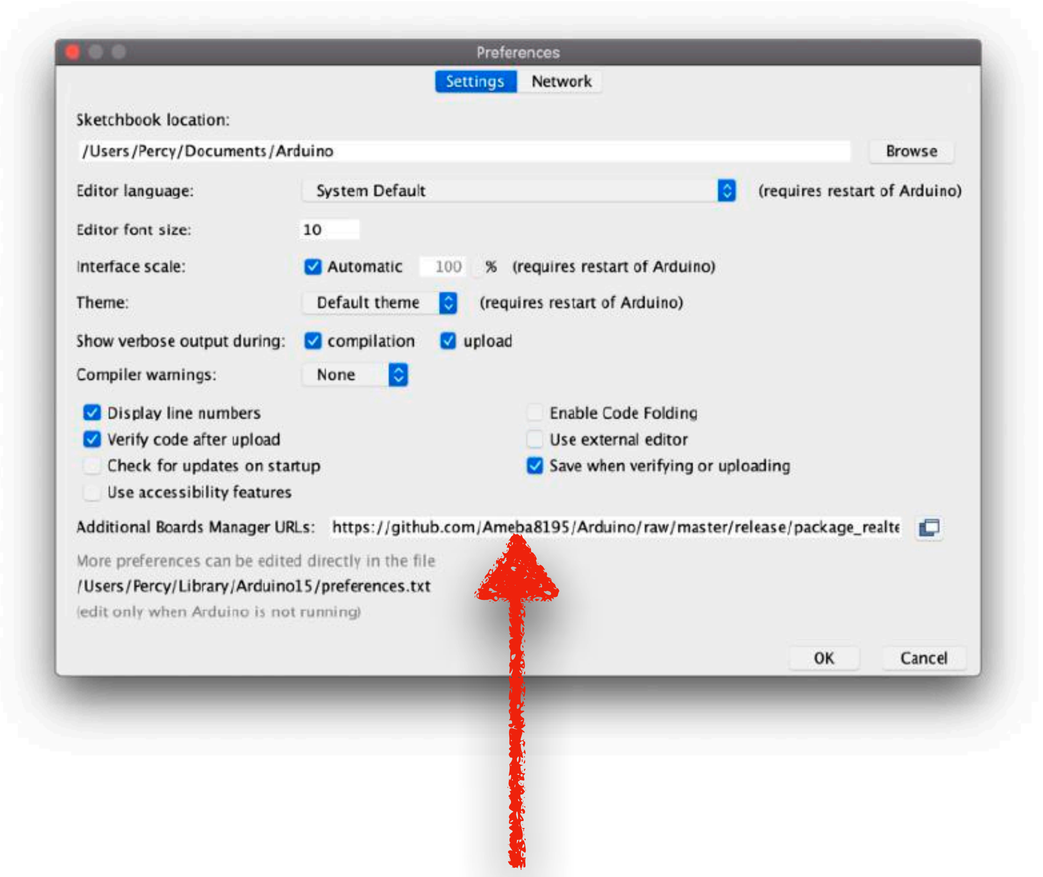
- Additional Arduino® integration add-ons will need to be installed.

    - Start your Arduino® IDE software.

    - Under the **File** menu, select **Preferences**.

    - Choose the **Settings** tab.

    - Add an entry into the **Additional Boards Manager URLs.**



Sketchbook location:
/Users/Percy/Documents/Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 10

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ☑ compilation ☑ upload

Compiler warnings: None

☑ Display line numbers
☑ Verify code after upload
□ Check for updates on startup
□ Use accessibility features
□ Enable Code Folding
□ Use external editor
☑ Save when verifying or uploading

Additional Boards Manager URLs: https://github.com/Ameba8195/Arduino/raw/master/release/package_realte

More preferences can be edited directly in the file
/Users/Percy/Library/Arduino15/preferences.txt
(edit only when Arduino is not running)

https://raw.githubusercontent.com/TomatoCube18/ch55xduino/ch55xduino/package_tc_ch55xduino_mcs51_index.json

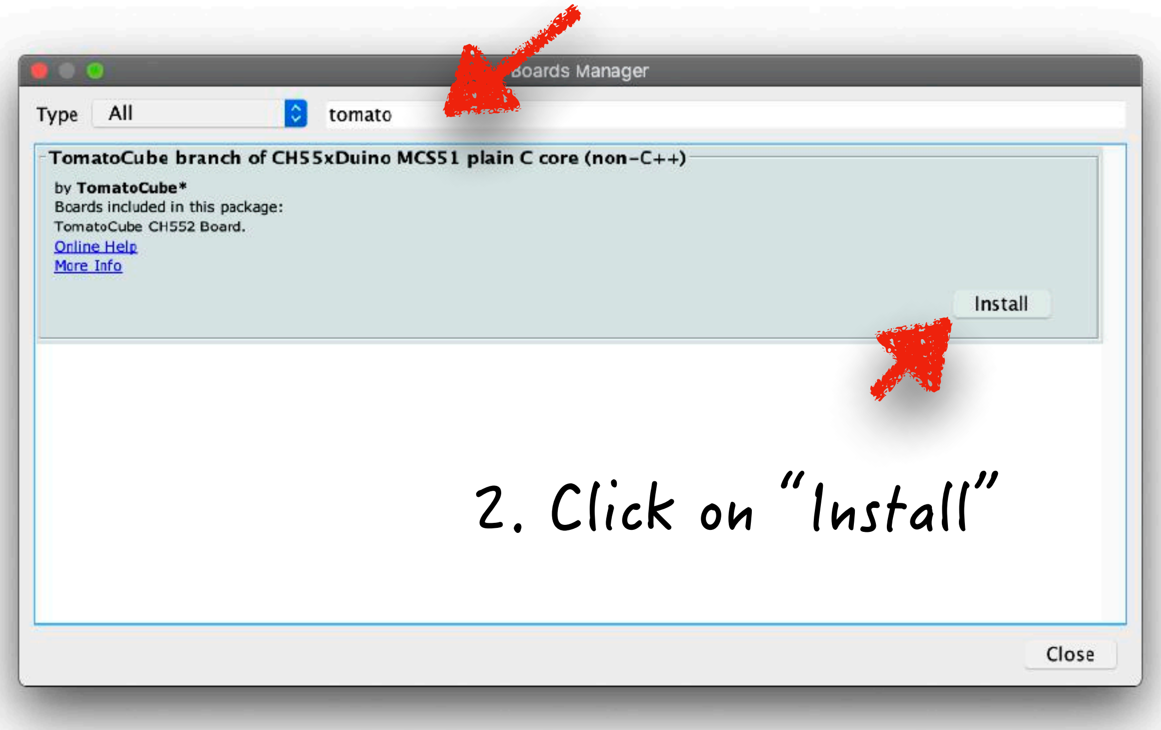
Adding the Board support using Arduino’s board manager

- Under the **Tools** menu, expand **Board: xxx** & choose **Boards Manager**.

- Reduce the list of entries by typing “**tomato**” into the textbox.

- Once installation of the new board is successful, you will find a new board entry being added to the list of boards.

1. Enter “Tomato” to filter the list of boards



Boards Manager

Type All tomato

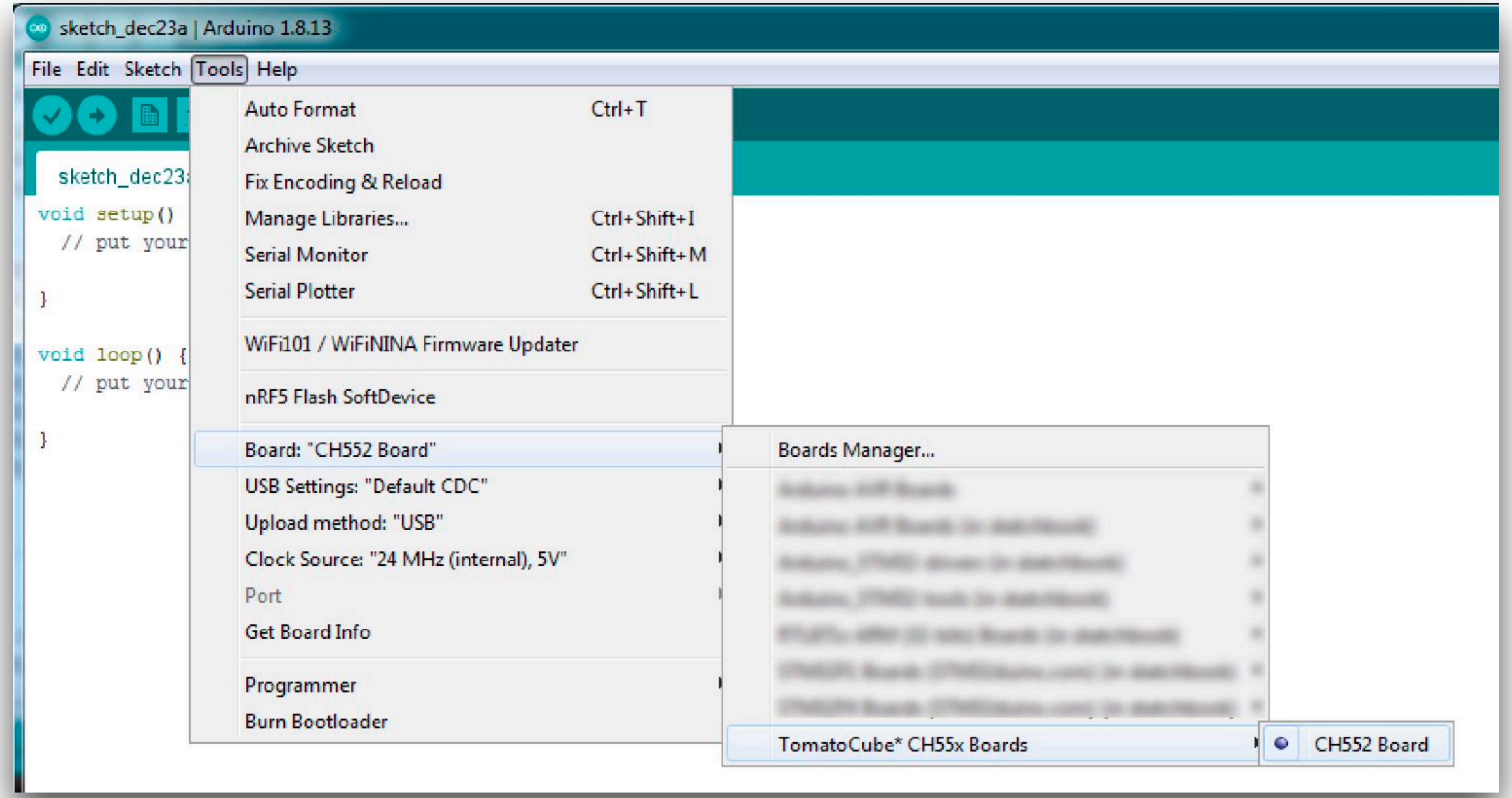
TomatoCube branch of CH55xDuino MCS51 plain C core (non-C++)

by **TomatoCube***
Boards included in this package:
TomatoCube CH552 Board.
Online Help
More Info

2. Click on “Install”

Choosing the correct board target

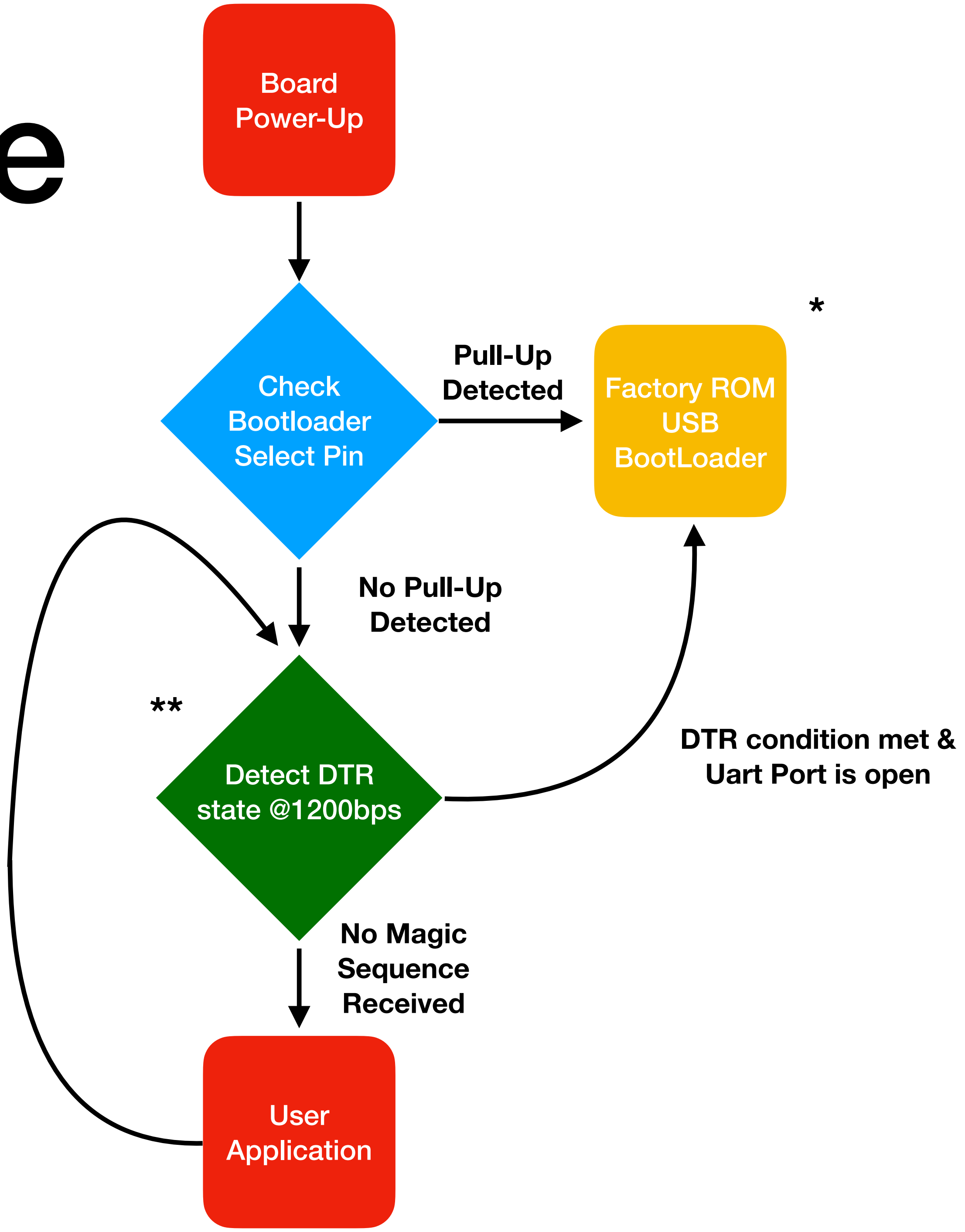
- Under the **Tools** menu, expand **Board: xxx** & choose **TomatoCube* CH55x Boards -> CH552 Board.**



```
void setup() {
  // put your

}

void loop() {
  // put your

}
```

Board: "CH552 Board"
USB Settings: "Default CDC"
Upload method: "USB"
Clock Source: "24 MHz (internal), 5V"

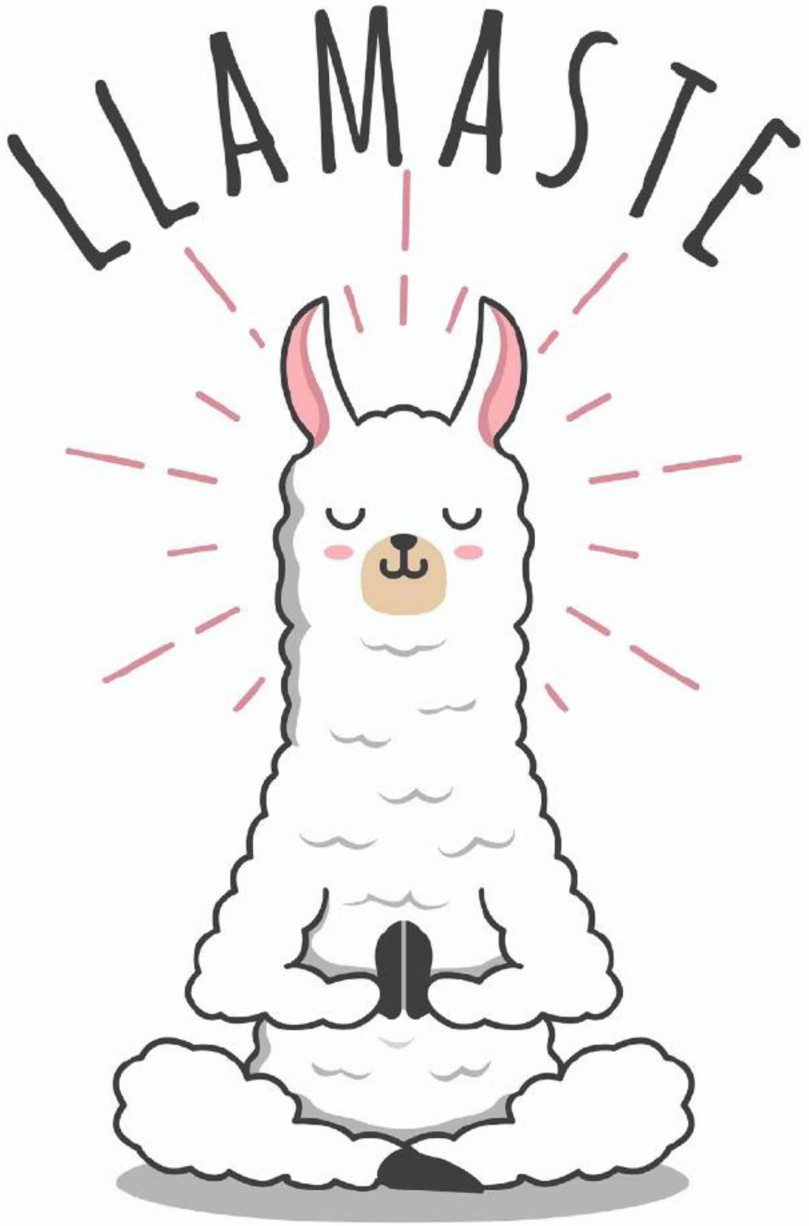
Happy CNY Card Boot-up Sequence (simplify)


Board Power-Up
Check Bootloader Select Pin
Pull-Up Detected
Factory ROM USB BootLoader
*
No Pull-Up Detected
**
Detect DTR state @1200bps
DTR condition met & Uart Port is open
No Magic Sequence Received
User Application

Zen…

While loading a new program (Sketch) to the board.

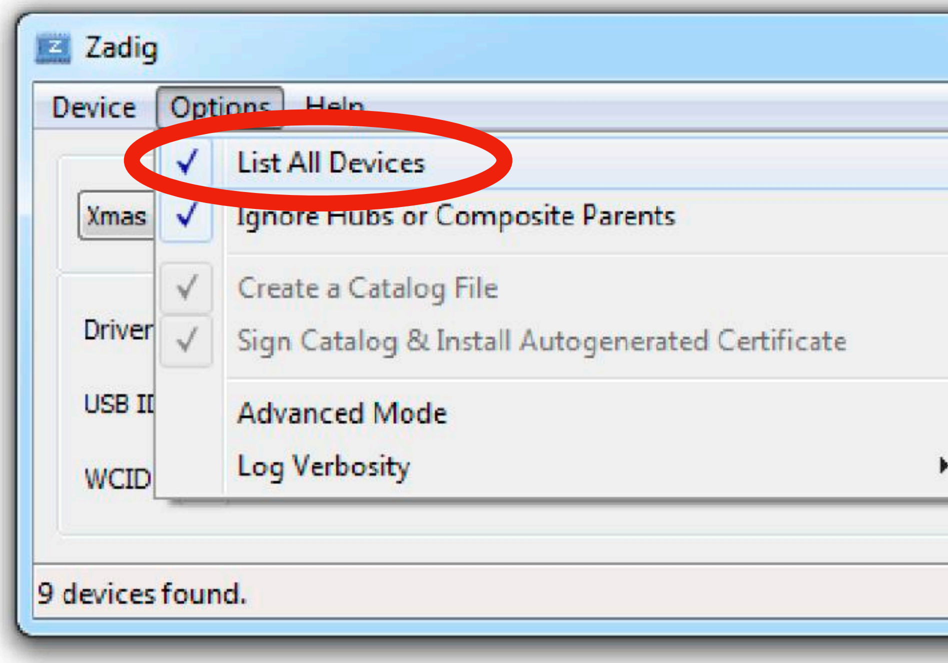
Let Arduino® IDE finish its task before unplugging.



If you ever corrupt the board's firmware,
you can refer to the appendix of this document
on how to rescue the board. :)

Microsoft® Windows device drivers (i)

If Device list is empty, check that "List ALL Devices" is selected



Zadig

Device | Options | Help

✓ List All Devices

✓ Ignore Hubs or Composite Parents

✓ Create a Catalog File

✓ Sign Catalog & Install Autogenerated Certificate

Advanced Mode

Log Verbosity

9 devices found.

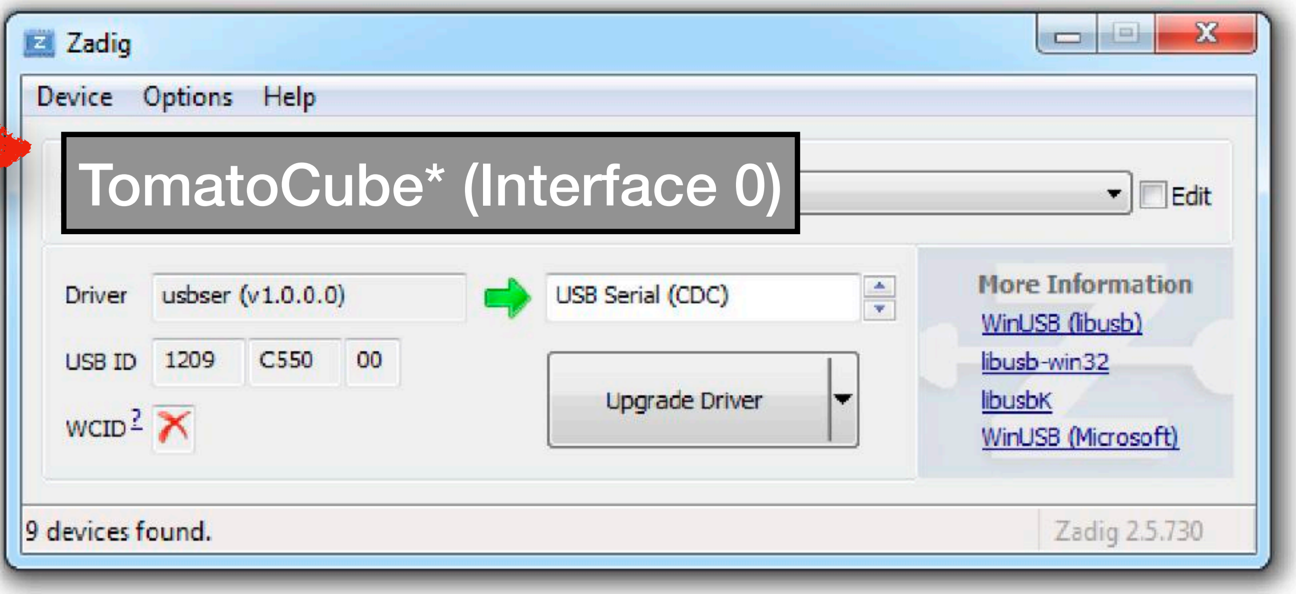
- We are using **Zadig** tool to setup (fix/replace) the windows drivers correctly. https://zadig.akeo.ie/

- The Happy CNY card will appear in two different mode.

We accidentally named some of the board as "X'mas 2020"

- Normal Operation [TomatoCube*] *USB Serial(CDC)*

- Boot-loader mode [USB Module] *libusb-win32 / WinUSB*



Zadig

TomatoCube* (Interface 0)

□ Edit

Driver: usbser (v1.0.0.0) → USB Serial (CDC)

USB ID: 1209 C550 00

WCID: ✗

9 devices found.

Zadig 2.5.730

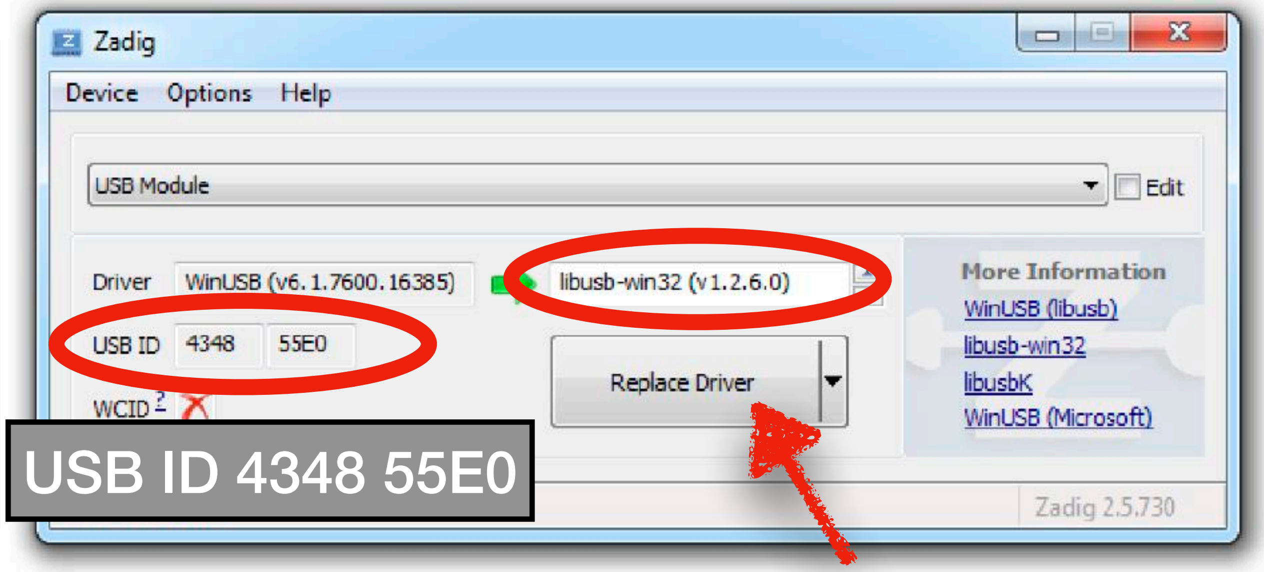
Microsoft® Windows device drivers (ii)

- Boot-loader mode
  [USB Module]
  libusb-win32 / *WinUSB (Not recommended)*

  - Arduino® will kick the CH552 board into Boot-loader when you try to upload a sketch.

    (Might need 2 tries/upload to get it to work).

  - After numerous test, we found that the Libusb-win32 driver to be far more reliable than Microsoft's **WinUSB** driver.



Zadig

USB Module ☐ Edit

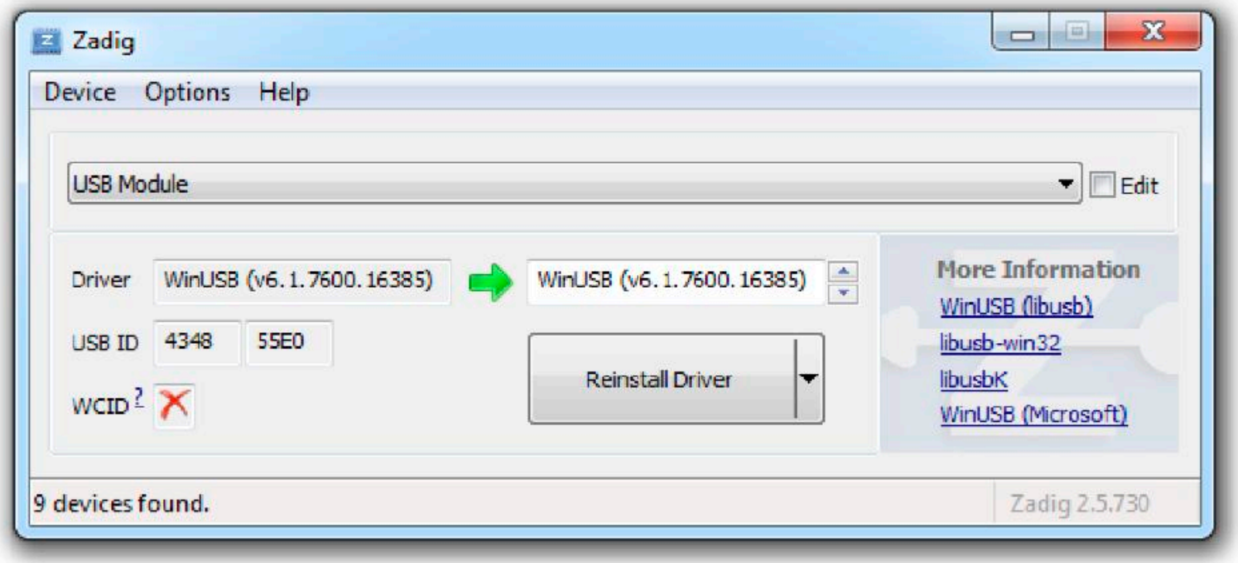
Driver WinUSB (v6.1.7600.16385) → libusb-win32 (v1.2.6.0)

USB ID 4348 55E0

WCID ✗

Zadig 2.5.730

USB ID 4348 55E0



Zadig

USB Module ☐ Edit

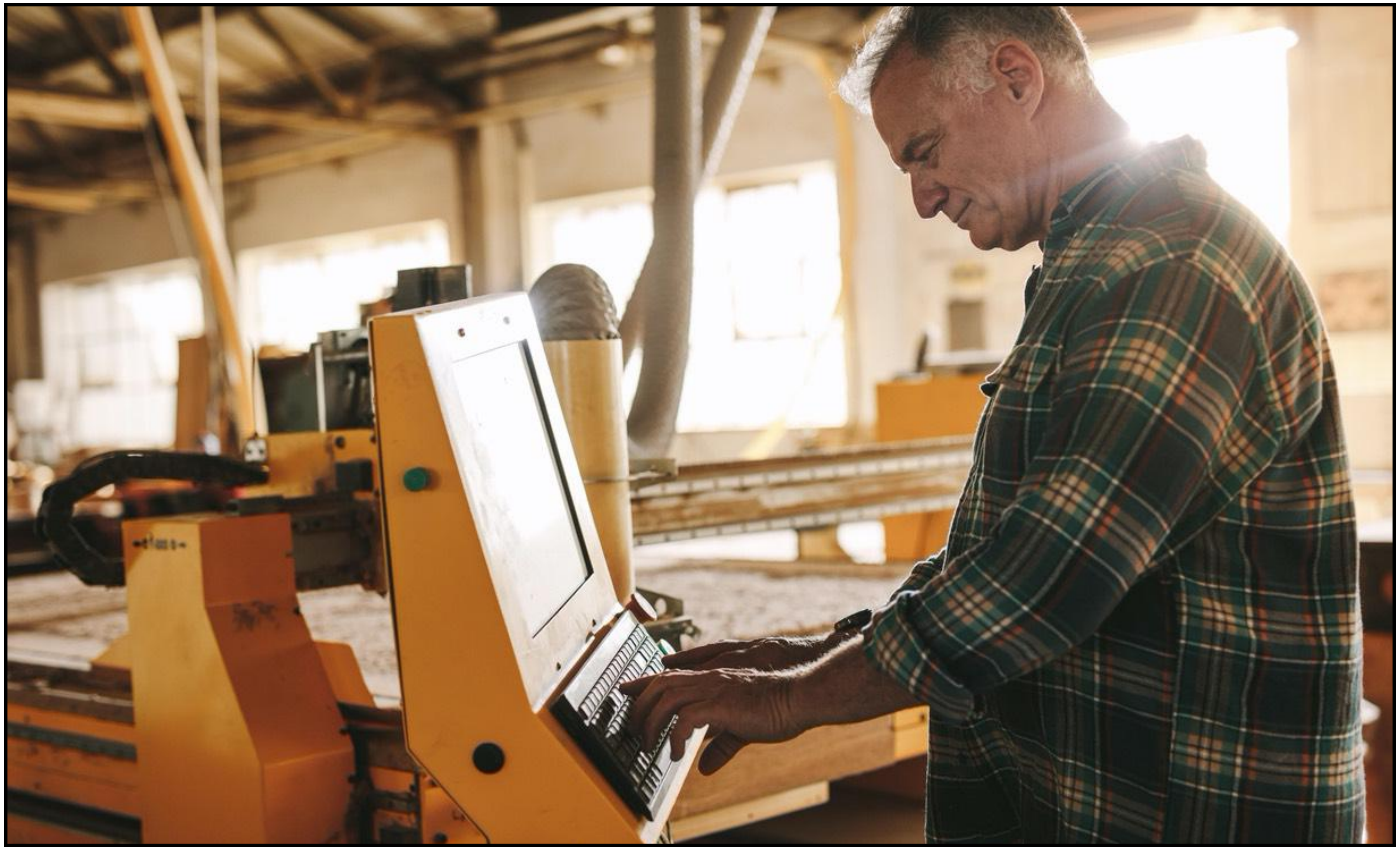
Driver WinUSB (v6.1.7600.16385) → WinUSB (v6.1.7600.16385)

USB ID 4348 55E0

WCID ✗

9 devices found.

Zadig 2.5.730



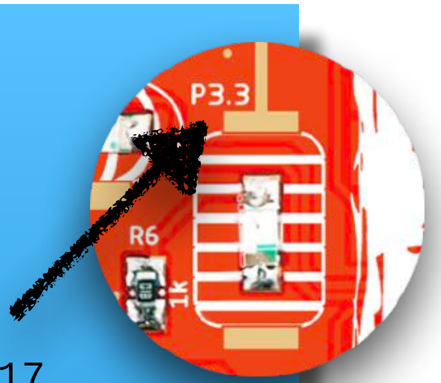
Male carpenter programming a cnc wood working machine in workshop by Jacob Lund from Noun Project

Coding

Quick & dirty intro to coding

Code 1 - Blinking Light (Digital Out)

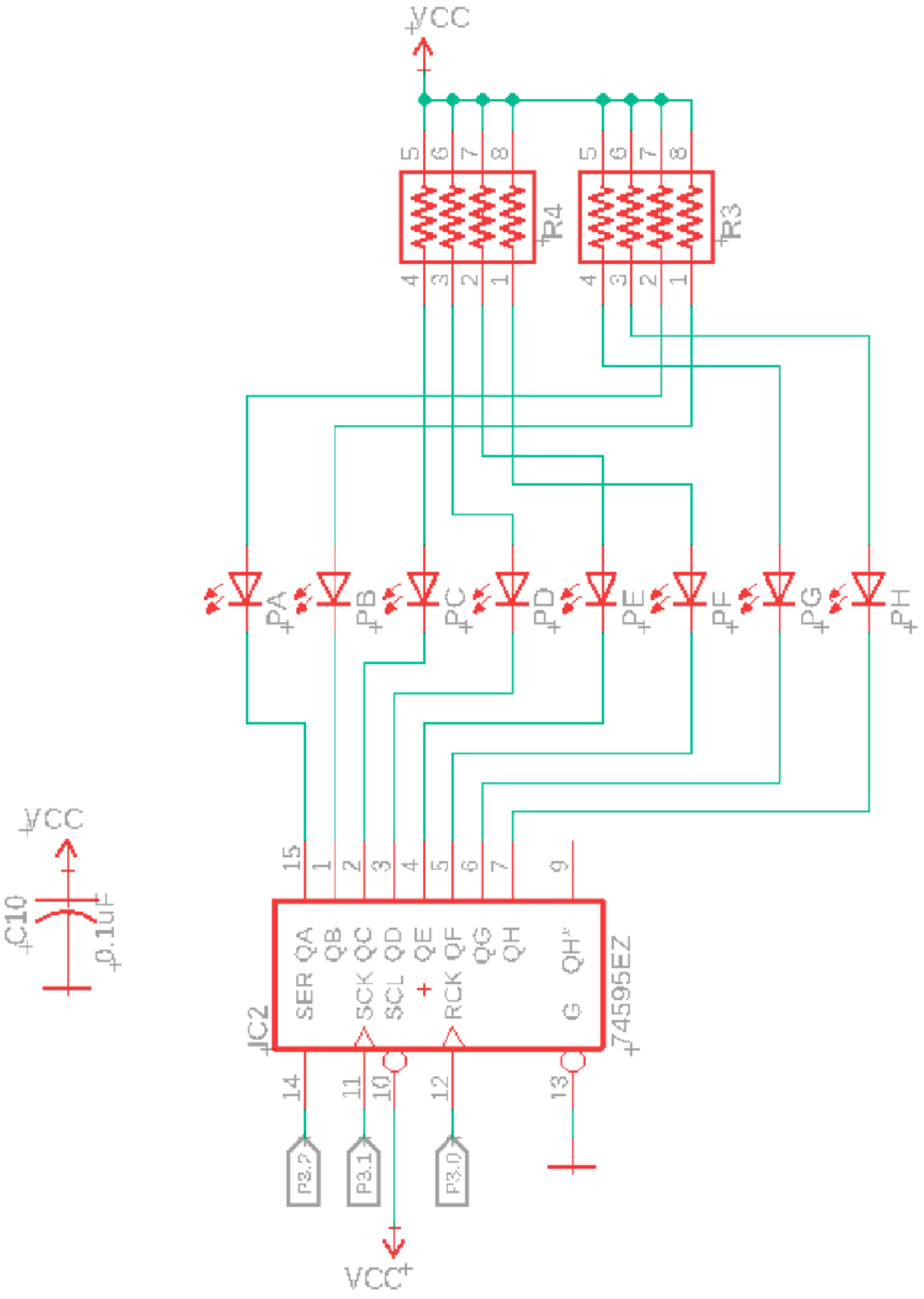
```
/*
 *   Ex_01 - DigitalOut
 *   Simple blinking of LED
 *
 */

#include "TomatoCubeWorker.h"
#define LED_BUILTIN 33    //other LEDs 30, 31, 32, 33, 16, 17

// the setup function runs once when you press reset or power the board
void setup() {
//  initTomatoCube();
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  // turn the LED on (HIGH is the voltage level)
  delay(250);
  // wait 250 milliseconds.
  digitalWrite(LED_BUILTIN, LOW);
  // turn the LED off by making the voltage LOW
  delay(500);
  // wait 500 milliseconds.
}
```



https://tomatocube.com/
/url/BQOEY

LEDs control circuit in V2 board (Shift-register)

- In order to allow our CPU (Brain) to control more digital output devices (LEDs).
    - Engineer will use either a port expander or a shift-register.
- Shift register ICs such as the 74LS595 acts like a mini memory cells.
- Using 3 digital pins from our CPU, we can control 8 individual LEDs.
    - Furthermore, the Shift register ICs could be cascaded.



Code 1.2 - Blinking Light
(Digital Out - Shift Register)

Pin definition of Shift-register Chip

Function to easily Control the LED Bits

```
/*
 *   Ex_01 - DigitalOut ShiftRegister
 *   Simple blinking of LED
 *
 */

#define Data_PINOUT   32
#define Clock_PINOUT  31
#define Latch_PINOUT  30

void shiftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder, uint8_t val);
void pixelLED(unsigned char ledPattern);

void shiftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder, uint8_t val)
{
     uint8_t i;
     for (i = 0; i < 8; i++)  {
           if (bitOrder == LSBFIRST)
                 digitalWrite(dataPin, !!(val & (1 << i)));
           else
                 digitalWrite(dataPin, !!(val & (1 << (7 - i))));
           digitalWrite(clockPin, HIGH);
           digitalWrite(clockPin, LOW);
     }
}

void pixelLED(unsigned char ledPattern) {
   digitalWrite(Latch_PINOUT, LOW);
   shiftOut(Data_PINOUT, Clock_PINOUT, LSBFIRST, ~ledPattern);
   digitalWrite(Latch_PINOUT, HIGH);
}
```

Code 1.2 - cont...

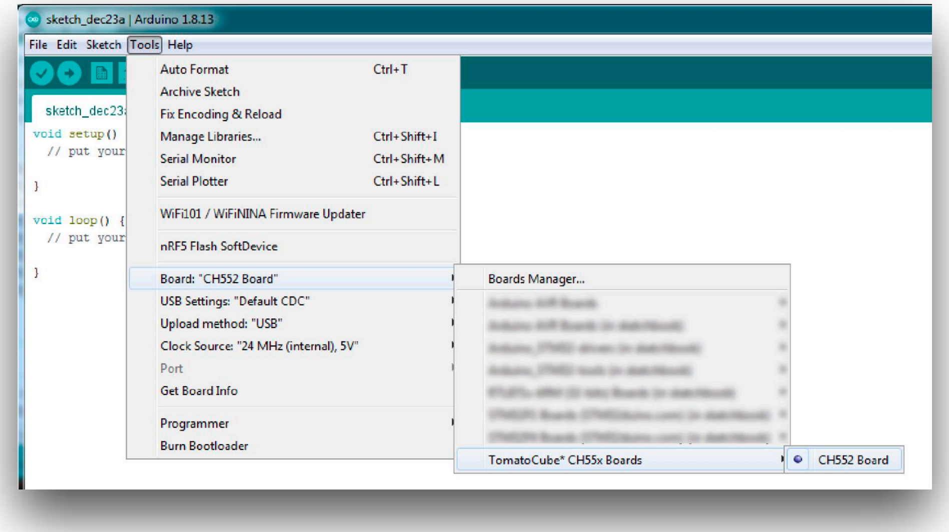
```
void setup() {
  // Shift register to control LED
  pinMode(Data_PINOUT, OUTPUT);
  pinMode(Clock_PINOUT, OUTPUT);
  pinMode(Latch_PINOUT, OUTPUT);
  digitalWrite(Data_PINOUT, LOW);
  digitalWrite(Clock_PINOUT, LOW);
  digitalWrite(Latch_PINOUT, LOW);
}

void loop() {
  pixelLED(0x01);
  // Light up the 1st LED
  delay(250);
  // wait for 250 milliseconds
  pixelLED(0x00);
  // Turn off all LEDs
  delay(250);
  // wait for 250 milliseconds

}
```

0x01 is a number represented in the Hexadecimal format
(We will discuss more in a later page)

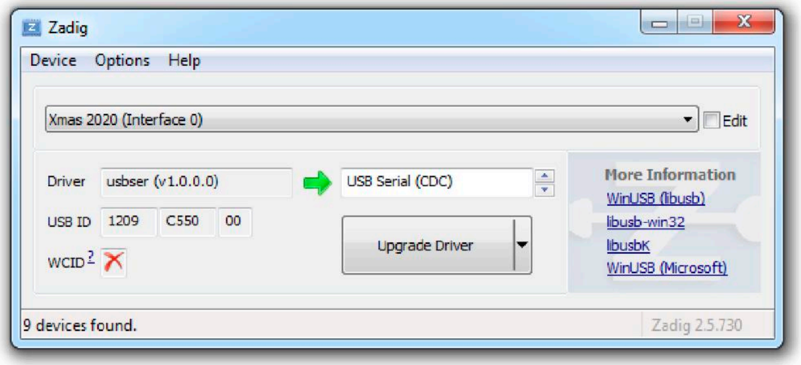
https://tomatocube.com//url/QR28Jg

Steps in uploading your Arduino® sketch to the board (1/2)

Step 1: Choosing the Right board (CH552 Board)



Step 2: Choosing the right driver USB Serial(CDC)



Xmas 2020 (Interface 0)

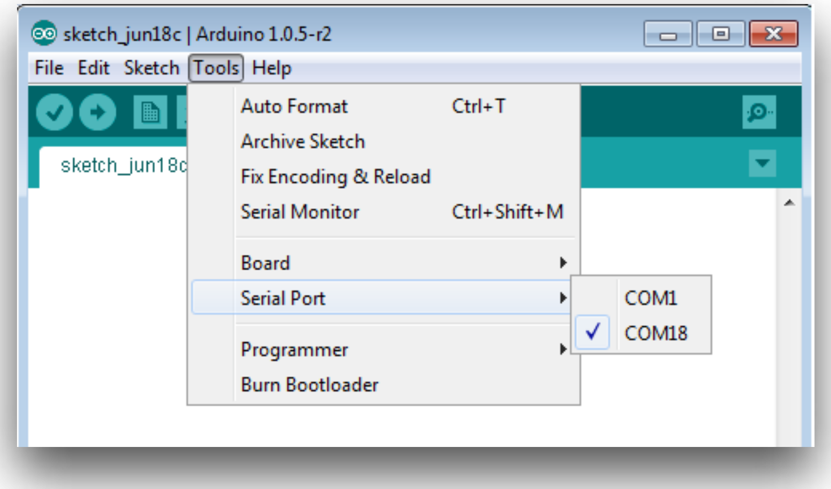
Driver usbser (v1.0.0.0) → USB Serial (CDC)

USB ID 1209 C550 00

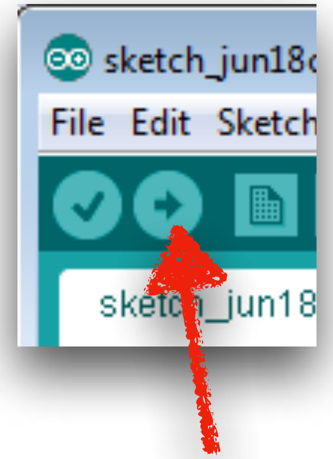
WCID ✗

9 devices found.

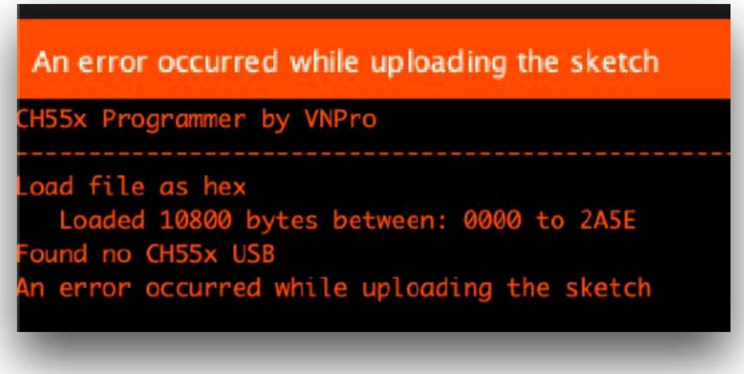
Step 3: Choosing the Right port



Step 4: Upload the Code



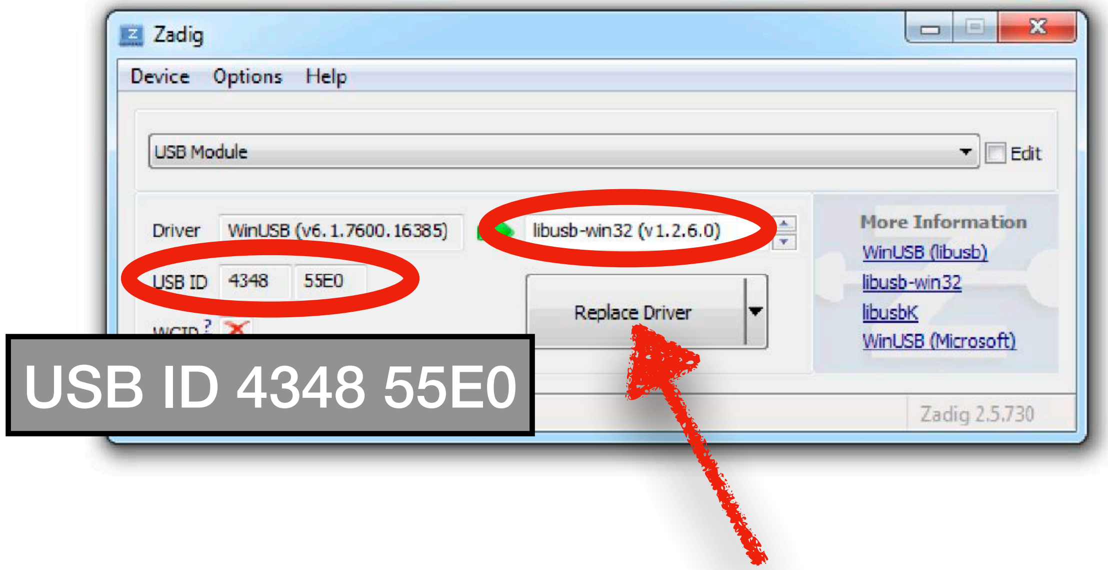
Step 5: The upload will fail for the first time(expected)



```
An error occurred while uploading the sketch
CH55x Programmer by VNPro
------------------------------------------------------------
Load file as hex
    Loaded 10800 bytes between: 0000 to 2A5E
Found no CH55x USB
An error occurred while uploading the sketch
```

Steps in uploading your Arduino® sketch to the board (2/2)

Step 6: Choosing the right driver libusb-win32



Zadig

Device Options Help

USB Module ☐ Edit

Driver WinUSB (v6.1.7600.16385) libusb-win32 (v1.2.6.0)

USB ID 4348 55E0

WCID ? ✗

Replace Driver

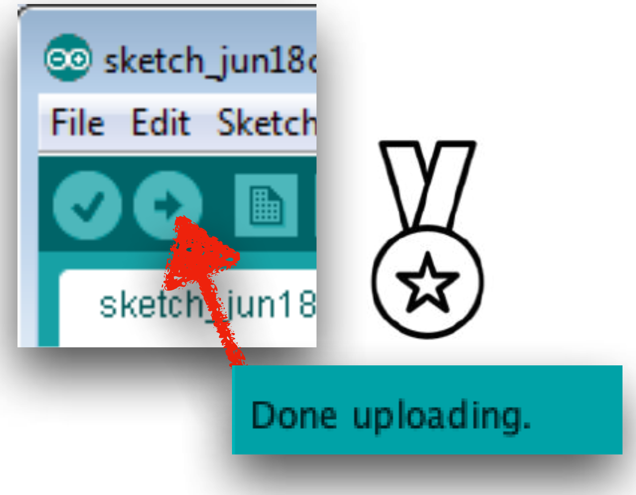
More Information
WinUSB (libusb)
libusb-win32
libusbK
WinUSB (Microsoft)

Zadig 2.5.730

USB ID 4348 55E0

Hit the "Replace Driver" Button

Step 7: Upload the code again



sketch_jun18c

File Edit Sketch

sketch_jun18

Done uploading.

Code 2 - Blinking Light revisited (Helper API & Binary Bit Shift)

computer or anything digital works
with Binary numbers. 0 & 1
our normal everyday numbers works
with Decimal system 0 to 10

<< means shifting by n number of bits

```
/*
 *   Ex_02 - HelperAPI
 *   Using helper API to easily initialize & control
 *   the Pixel LEDs
 */

#include "TomatoCubeWorker.h"

void setup() {
  initTomatoCube();

}

void loop() {
  pixelLED(0);
  delay(250);
  pixelLED(1);
  delay(250);
  pixelLED(1 << 1);    // Binary Number system
  delay(250);
  pixelLED(1 << 2);
  delay(250);
  pixelLED(1 << 3);
  delay(250);
  pixelLED(1 << 4);
  delay(250);
  pixelLED(1 << 5);
  delay(250);
}
```

PixelLED(binaryLEDPattern)

e.g. 1 << 3 = 0b 00 1000

https://tomatocube.com/url/QJyYO

Code 2.2 - Blinking Light revisited (Helper API & Binary Bit Shift)

computer or anything digital works with Binary numbers. 0 & 1
our normal everyday numbers works with Decimal system 0 to 10

<< means shifting by n number of bits

```
/*
 *  Ex_02.2 - HelperAPI
 *  Using helper API to easily initialize & control
 *  the Pixel LEDs
 */

#include "TomatoCubeWorker_v2.h"

void setup() {
  initTomatoCube();

}

void loop() {
  pixelLED(1);
  delay(250);
  pixelLED(1 << 1);
  delay(250);
  pixelLED(1 << 2);
  delay(250);
  pixelLED(1 << 3);
  delay(250);
  pixelLED(1 << 4);
  delay(250);
  pixelLED(1 << 5);
  delay(250);
  pixelLED(1 << 6);
  delay(250);
  pixelLED(1 << 7);
  delay(250);
}
```

Moving forward, replace the header file from **TomatoCubeWorker.h** to **TomatoCubeWorker_v2.h** yourself !

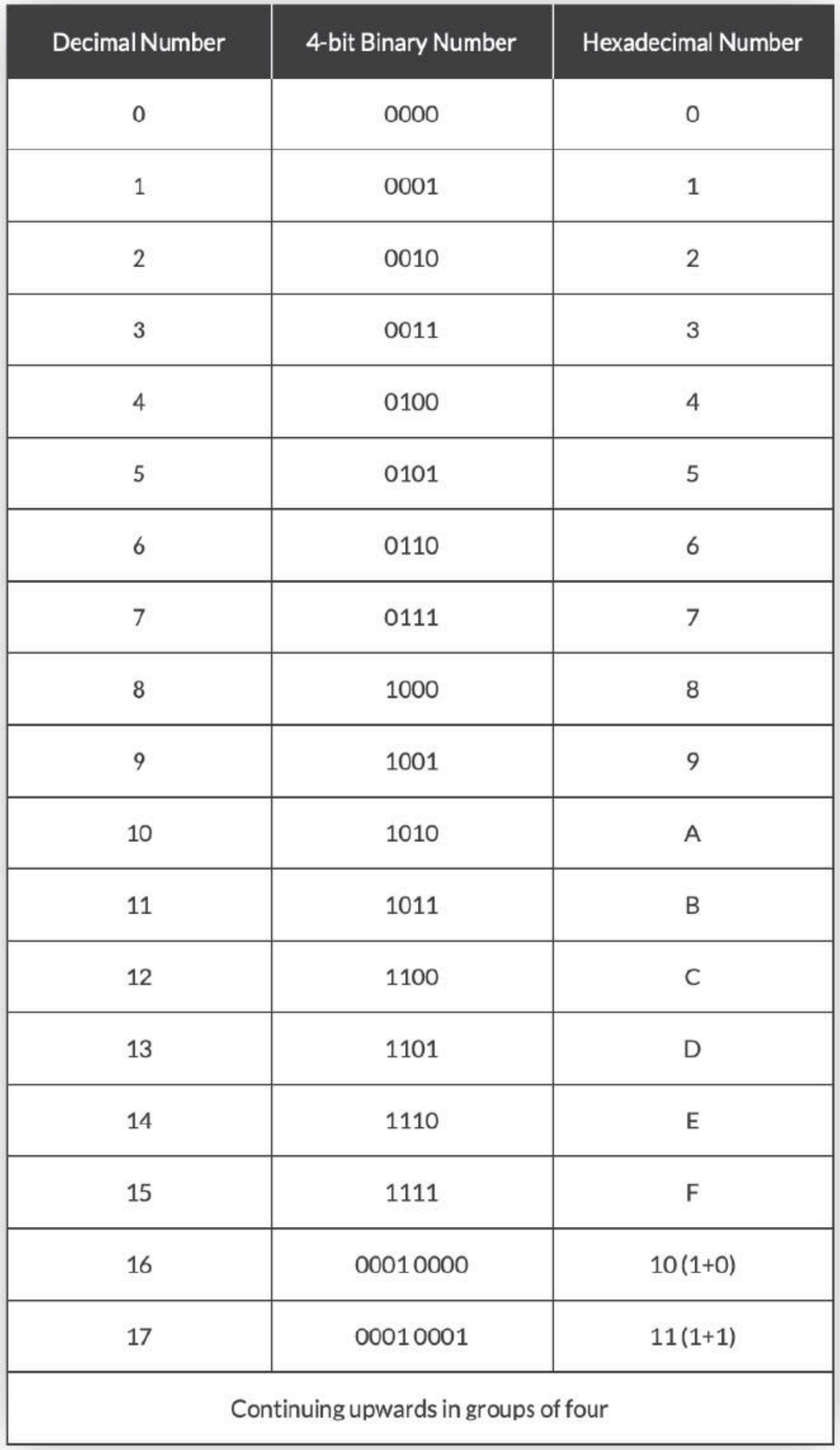
PixelLED(binaryLEDPattern**)**

e.g. 1 << 3 = 0b 00 1000

https://tomatocube.com/url/wnkNB

Number System

- **Decimal**
  - Decimal (base 10) system, and are very comfortable for human to perform operations with, it is also important for us to understand that the decimal system is not the only system in the world.
- **Binary**
  - A Binary number system has only two digits that are 0 and 1. Every digit (number) represents with 0 and 1 in this number system. The base of binary number system is 2, because it has only two digits.
- **Hexadecimal**
  - A Hexadecimal number system has sixteen alphanumeric values from 0 to 9 and A to F. The base of hexadecimal number system is 16, because it has 16 alphanumeric values.



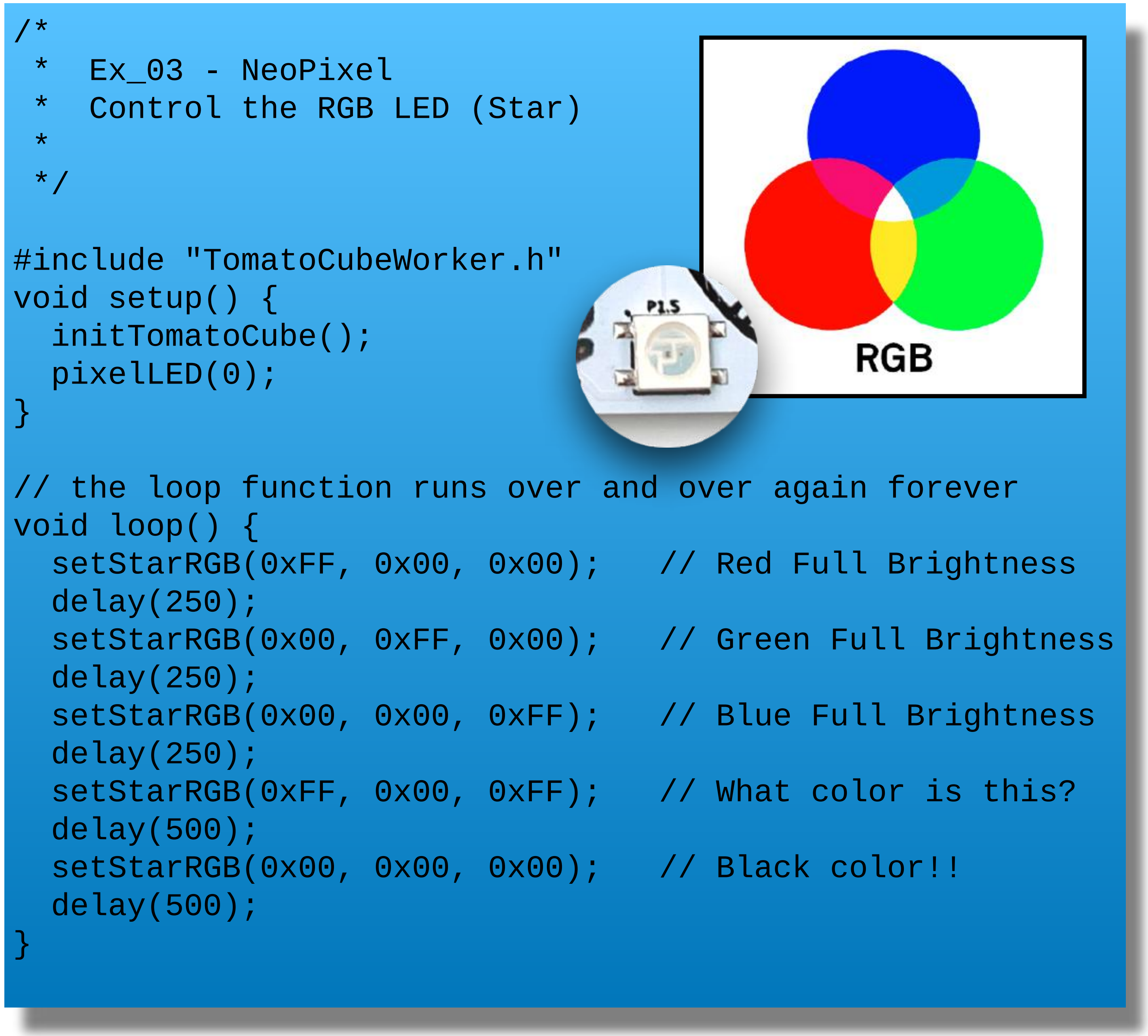
Decimal Number	4-bit Binary Number	Hexadecimal Number
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10 (1+0)
17	0001 0001	11 (1+1)
Continuing upwards in groups of four		

Code 3 - Mixing LED color (NeoPixel RGB)

And now we are going to use hexadecimal number system. 0 to 15 or 0~9,A~F

Each color is represented by 8 bits, thus it goes from 0X00 to 0XFF

https://tomatocube.com/url/wnkAd



```
/*
 *  Ex_03 - NeoPixel
 *  Control the RGB LED (Star)
 *
 */

#include "TomatoCubeWorker.h"
void setup() {
  initTomatoCube();
  pixelLED(0);
}

// the loop function runs over and over again forever
void loop() {
  setStarRGB(0xFF, 0x00, 0x00);   // Red Full Brightness
  delay(250);
  setStarRGB(0x00, 0xFF, 0x00);   // Green Full Brightness
  delay(250);
  setStarRGB(0x00, 0x00, 0xFF);   // Blue Full Brightness
  delay(250);
  setStarRGB(0xFF, 0x00, 0xFF);   // What color is this?
  delay(500);
  setStarRGB(0x00, 0x00, 0x00);   // Black color!!
  delay(500);
}
```

Basic Programing Construct (i): Conditional

- **Conditionals** are found in all forms of programming.

```
if (condition) {}
else if (another condition) {}
else {}
```

- They allow our program to react to conditions and make a choice from one or many choices.

Code 4 - Touch inputs (Capacitive touch sensing & conditional)

https://tomatocube.com/url/d23V6

```
/*
 *   Ex_04 - TouchInput
 *   Sample 2 touch inputs P1.1 & P1.4
 *
 *
 */

#include "TomatoCubeWorker.h"

void setup() {
  initTomatoCube();
  pixelLED(0);
  setStarRGB(0x00, 0x00, 0x00);
}

// the loop function runs over and over again forever
void loop() {
  scanTouchButton();
  if (getTouchB1Transition()) {
      setStarRGB(0xFF, 0x00, 0x00);
  }

  else if (getTouchB2Transition()) {
      setStarRGB(0x00, 0xFF, 0x00);
  }

}
```

Basic Programing Construct (ii): Variable

- **Variables** are core fundamental to any programming language.

- They are small chunk of **memory** within your program, or you can think of it like **boxes**.

Code 5 - Learn about “Variables” (Variables & C Built-in functions)

```
/*
 *   Ex_05 - TouchInput
 *   Sample 2 touch inputs P1.1 & P1.4
 *   P1.1 will be checking the transition (onTouch).
 *   p1.4 will be checking the transition (onTouch).
 *   And we are introducing the concept of variables.
 *
 */

#include "TomatoCubeWorker.h"
#include <math.h>

// Variable to fold the state of the LED.
int ledPixel = 0;

void setup() {
  initTomatoCube();
  pixelLED(1);
  setStarRGB(0x00, 0x00, 0x00);
}
```

Variable of type integer

`powf(x, n)`

means x to the power of n.

`rand()`

Random number generation.

```
// the loop function runs over and over again forever
void loop() {
  scanTouchButton();

  if (getTouchB1Transition()) {
      ledPixel++;
      if (ledPixel > 5) {   // Change 5 to 7 for v2
          ledPixel = 0;
      }
      pixelLED(powf(2, ledPixel));  // 2 ^ ledPixel
  }

  if (getTouchB2Transition()) {
      pixelLED(rand() % 0x3F);// Change 3F to FF for v2
      // Generate a random pattern between 0 ~ 0x3f
  }

}
```

https://tomatocube.com/url/nevE3

Code 6 - Making Sound (Generating sound frequency)

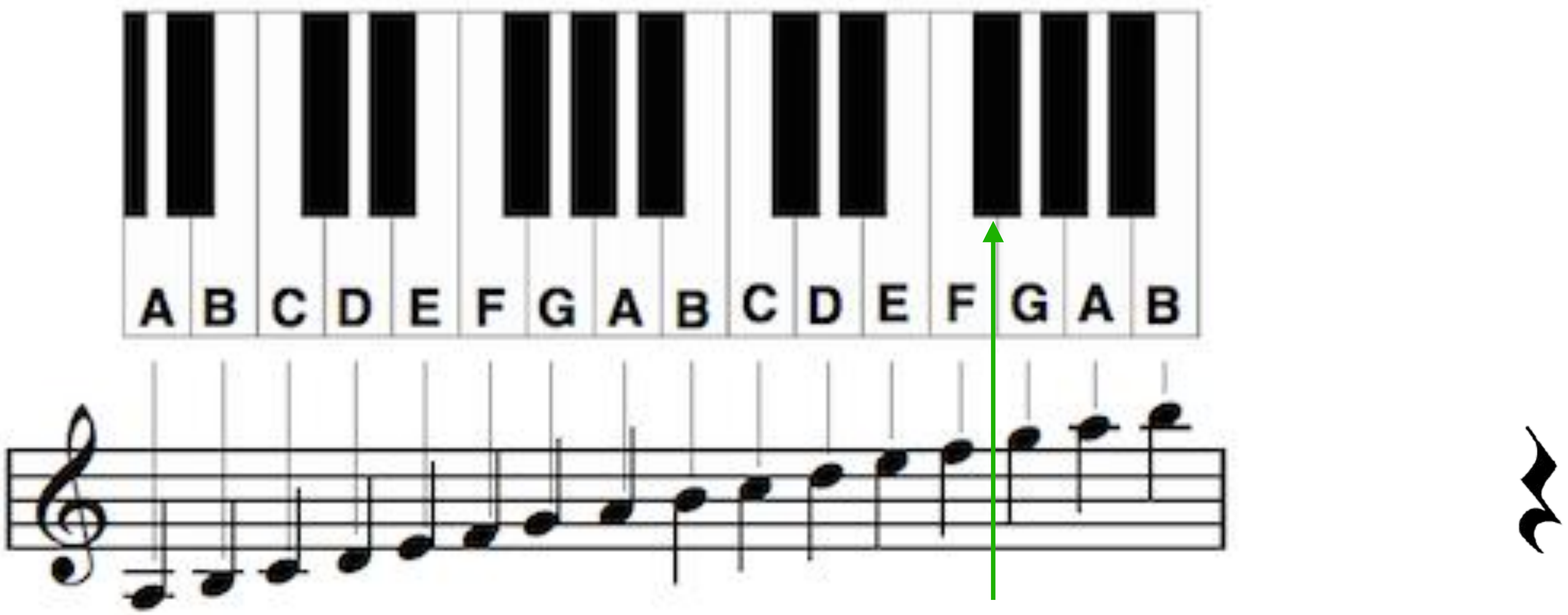
https://tomatocube.com/url/9JEn9

```
/*
 *  Ex_06 - Tone
 *  Create Tone for BabyShark & StarWars
 *
 */

#include "TomatoCubeWorker.h"


void setup() {
  initTomatoCube();
  pixelLED(0);
  setStarRGB(0x00, 0x00, 0x00);
}

// the loop function runs over and over again forever
void loop() {
    scanTouchButton();
    if (getTouchB1Transition()) {
        //BabyShark Theme
        playTone(TONE_PINOUT, NOTE_D5, 400);
        playTone(TONE_PINOUT, REST, 10);
        playTone(TONE_PINOUT, NOTE_E5, 400);
         // ...
        playTone(TONE_PINOUT, REST, 20);
    }

    if (getTouchB2Transition()) {
        //StarWars Theme
        playTone(TONE_PINOUT, NOTE_A5, 500);
        playTone(TONE_PINOUT, NOTE_A5, 500);
        playTone(TONE_PINOUT, NOTE_A5, 500);
          // ...
        playTone(TONE_PINOUT, REST, 350);

    }

}
```

Music Note

milliseconds

About the Notes



NOTE_C2 ... NOTE_C4 ... NOTE_A4 NOTE_B4 NOTE_C5 ... NOTE_FS5 ... NOTE_B6 REST

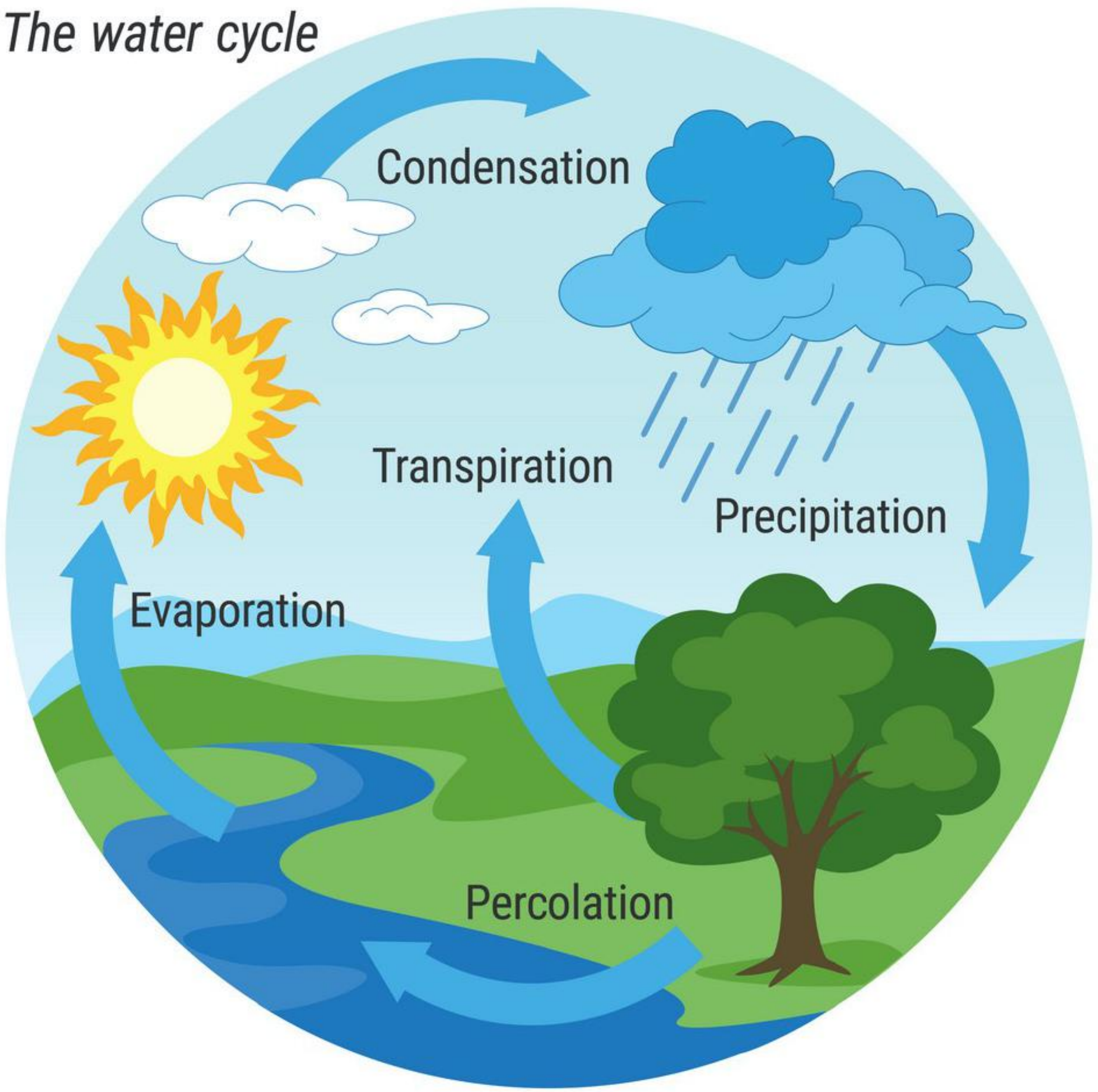
F#

Basic Programing Construct (iii): Loops

- **Loops** are yet another programming construct found in all forms of programming.

- In the world of C programming, there are **three** different types of Loops at your disposal.
  - `for loop`
  - `while loop`
  - `do…while loop`

- They allow our program to repeat a chunk of tasks repetitively



Code 7 - Repetitive process (Loops & subroutine)

```
/*
 *  Ex_07 - Loops & subroutine
 *  Repeating code a number of time &
 *  Abstrating code into smaller chunk
 *  subroutine
 *
 */

#include "TomatoCubeWorker.h"

// Variable to fold the state of the LED.
unsigned char randomLEDSeq[20];

void generateLEDSequence() {
    srand(millis());
    for (int i = 0; i < 20; i ++) {
        randomLEDSeq[i] = rand() % 0x3F;
        // change 3F to FF for v2 board
    }
}

unsigned char getRandomLED() {
    static unsigned char currentLEDPattern = 0;
    currentLEDPattern += 1;
    if (currentLEDPattern >= 20)
        currentLEDPattern = 0;
    return randomLEDSeq[currentLEDPattern];
}
```

Arrays

```
void setup() {
  initTomatoCube();
  pixelLED(0);
  setStarRGB(0x00, 0x00, 0x00);
  generateLEDSequence();
}

// the loop function runs over and over again forever
void loop() {
  pixelLED(getRandomLED());
  delay(250);
}
```

Arrays are a collection of the same type of variables.

https://tomatocube.com/url/8R6px

Code 8 - Code tidying & better Music code (Using Loops)

```
/*
 *  Ex_08 - BetterTone
 *  Better way of playing a tune
 *
 */

#include "TomatoCubeWorker.h"
const PROGMEM char cnySong[] = {
  NOTE_D4, 8, NOTE_E4, 8, NOTE_F4, 8, NOTE_G4, 8, NOTE_AS4, 4, NOTE_A4,  4,
  NOTE_A4, 8, NOTE_D5, 8, NOTE_D5, 8, NOTE_A4, 8, NOTE_A4, 4, NOTE_G4,  4,
  NOTE_G4, 8, NOTE_AS4, 8, NOTE_A4, 8, NOTE_G4, 8, NOTE_G4, 4, NOTE_F4,  4,
  NOTE_F4, 8, NOTE_E4, 8, NOTE_D4, 8, NOTE_CS4, 8, NOTE_D4, 4, NOTE_D4,  4,
  NOTE_G4, 136, NOTE_A4, 16, NOTE_F4, 136, NOTE_A4, 16, NOTE_E4, 8, NOTE_A4,  8,
  NOTE_D4, 8, NOTE_A4,  8, NOTE_G4, 136, NOTE_A4, 16, NOTE_F4, 136, NOTE_A4, 16,
  NOTE_E4, 8, NOTE_A4,  8, NOTE_D4, 4, NOTE_G4, 136, NOTE_A4, 16, NOTE_F4, 136,
  NOTE_A4, 16, NOTE_E4, 8, NOTE_A4,  8, NOTE_D4, 8, NOTE_A4,  8, NOTE_G4, 136,
  NOTE_A4, 16, NOTE_F4, 136, NOTE_A4, 16, NOTE_E4, 8, NOTE_A4,  8, NOTE_D4, 4
};


void setup() {
  initTomatoCube();
  pixelLED(0);
  setStarRGB(0x00, 0x00, 0x00);
}
```

https://tomatocube.com/url/EPXaKB

Code 8 - cont…

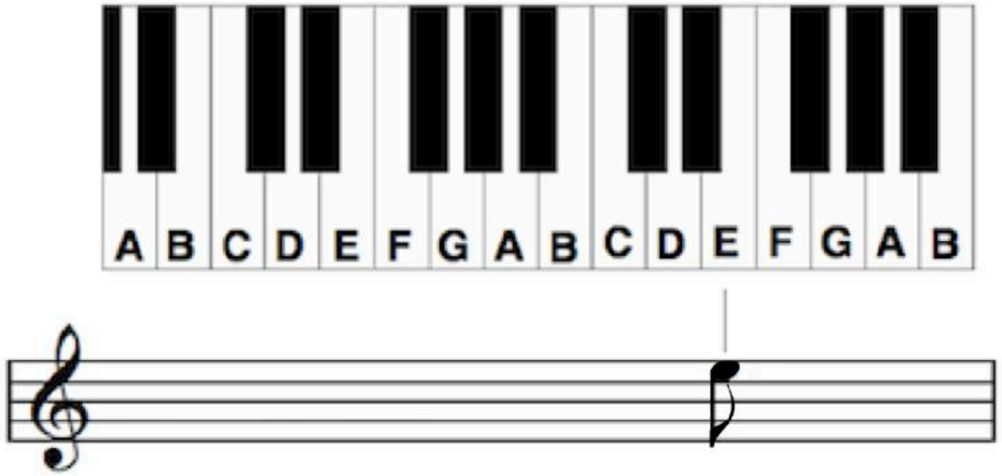
```
// the loop function runs over and over again forever
void loop() {
     scanTouchButton();
     if (getTouchB1Transition()) {
          char *songNotes = cnySong;
          unsigned char size = sizeof(cnySong);
          size = (size/sizeof(unsigned char))/2;  // Number of music notes (freq + duration)

          for (int thisNote = 0; thisNote < size; thisNote++) {

            int noteDuration = 0;
            if (songNotes[(thisNote * 2) + 1] < 128) {
              // regular note, just proceed
              noteDuration = (getWholeNote()) / (songNotes[(thisNote * 2) + 1]);
            } else {
              noteDuration = (getWholeNote()) / ((songNotes[(thisNote * 2) + 1]) - 128);
              noteDuration *= 1.5; // increases the duration in half for dotted notes
            }

            playTone(TONE_PINOUT, songNotes[(thisNote * 2)], noteDuration);

            // to distinguish the notes, set a minimum time between them.
            int pauseBetweenNotes = noteDuration * 1.30;
            delay(pauseBetweenNotes);

          }
          playTone(TONE_PINOUT, REST, 100);
     }
}
```

Notes entry format

Note	Name	Beats
𝅝	Whole note	4 beats
𝅗𝅥	Half note	2 beats
♩	Quarter note	1 beat
♪	Eighth note	½ beat
𝅘𝅥𝅯	Sixteenth note	¼ beat

```
E Note from the 5th octave,
for 1/8 duration of a whole note.
(quaver)
```

```
NOTE_E5, 8
```



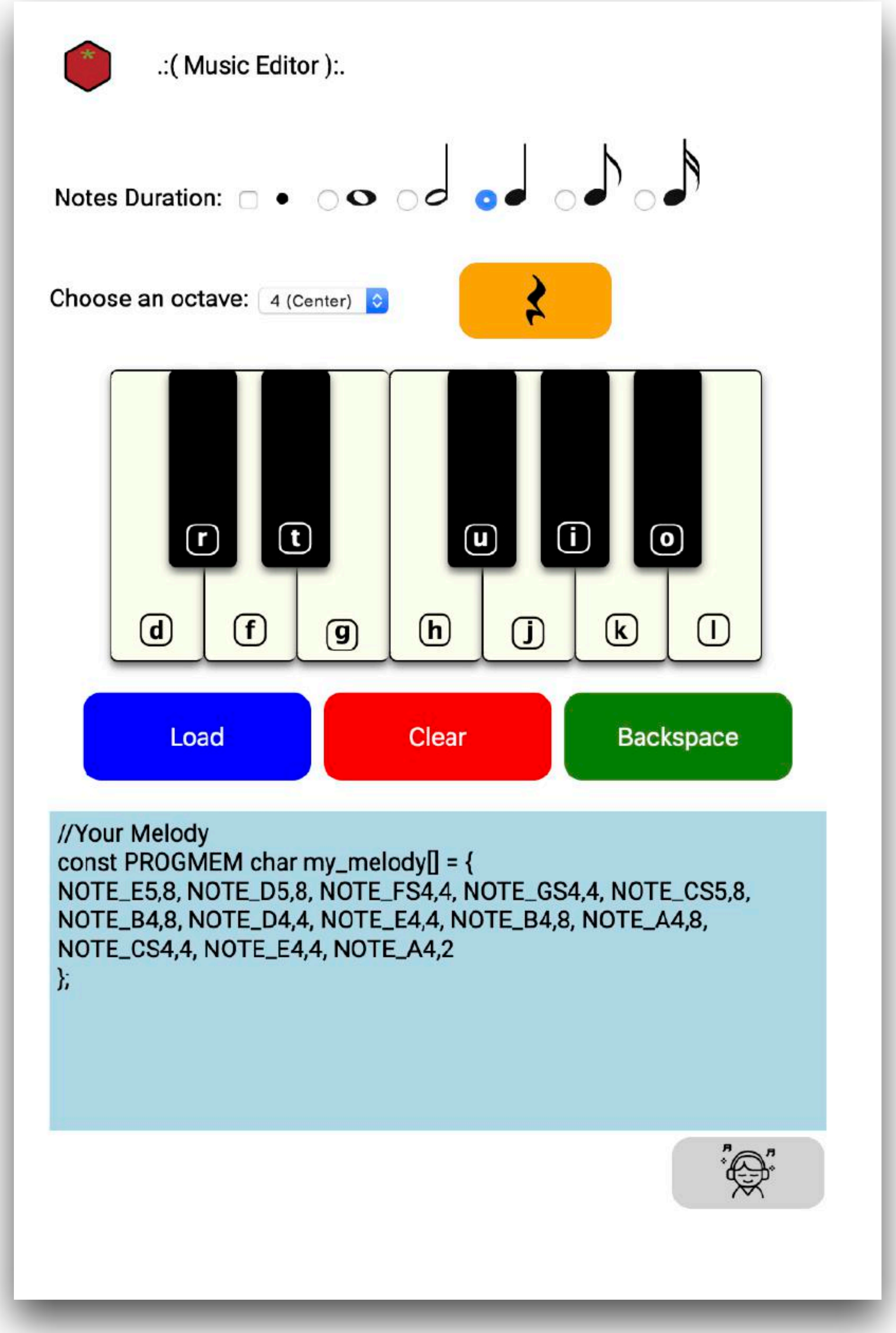
For dotted note, add 0x80 (128)
to the notes representation.

e.g. dotted crochet = 4 + 128 = 132
dotted quaver = 8 + 128 = 136

Making your life easier

Music editor for those that would like to take up the challenge.

https://tomatocube.com/musicEditor/

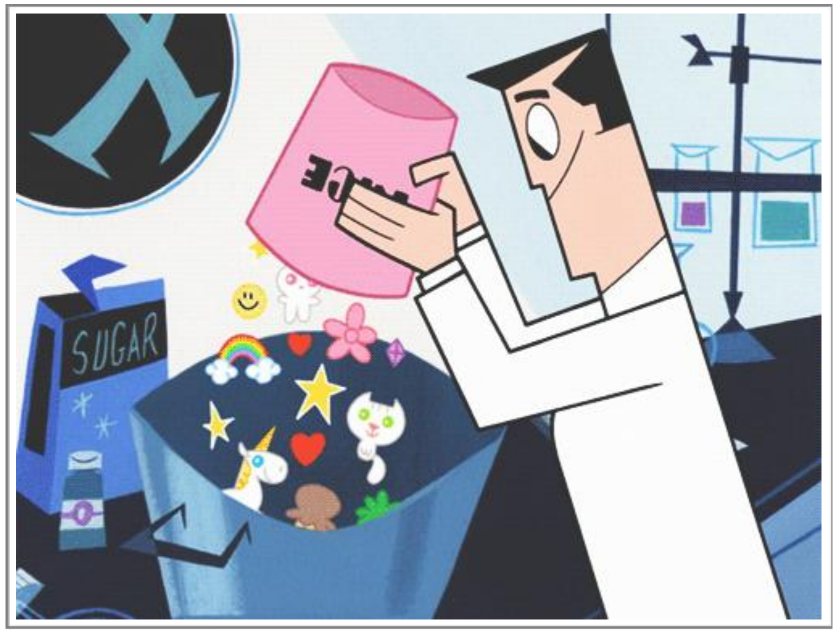


Choose an octave: 4 (Center)

```
//Your Melody
const PROGMEM char my_melody[] = {
NOTE_E5,8, NOTE_D5,8, NOTE_FS4,4, NOTE_GS4,4, NOTE_CS5,8,
NOTE_B4,8, NOTE_D4,4, NOTE_E4,4, NOTE_B4,8, NOTE_A4,8,
NOTE_CS4,4, NOTE_E4,4, NOTE_A4,2
};
```

Code 9 - Mixing it together (Combining all the previous code & techniques)

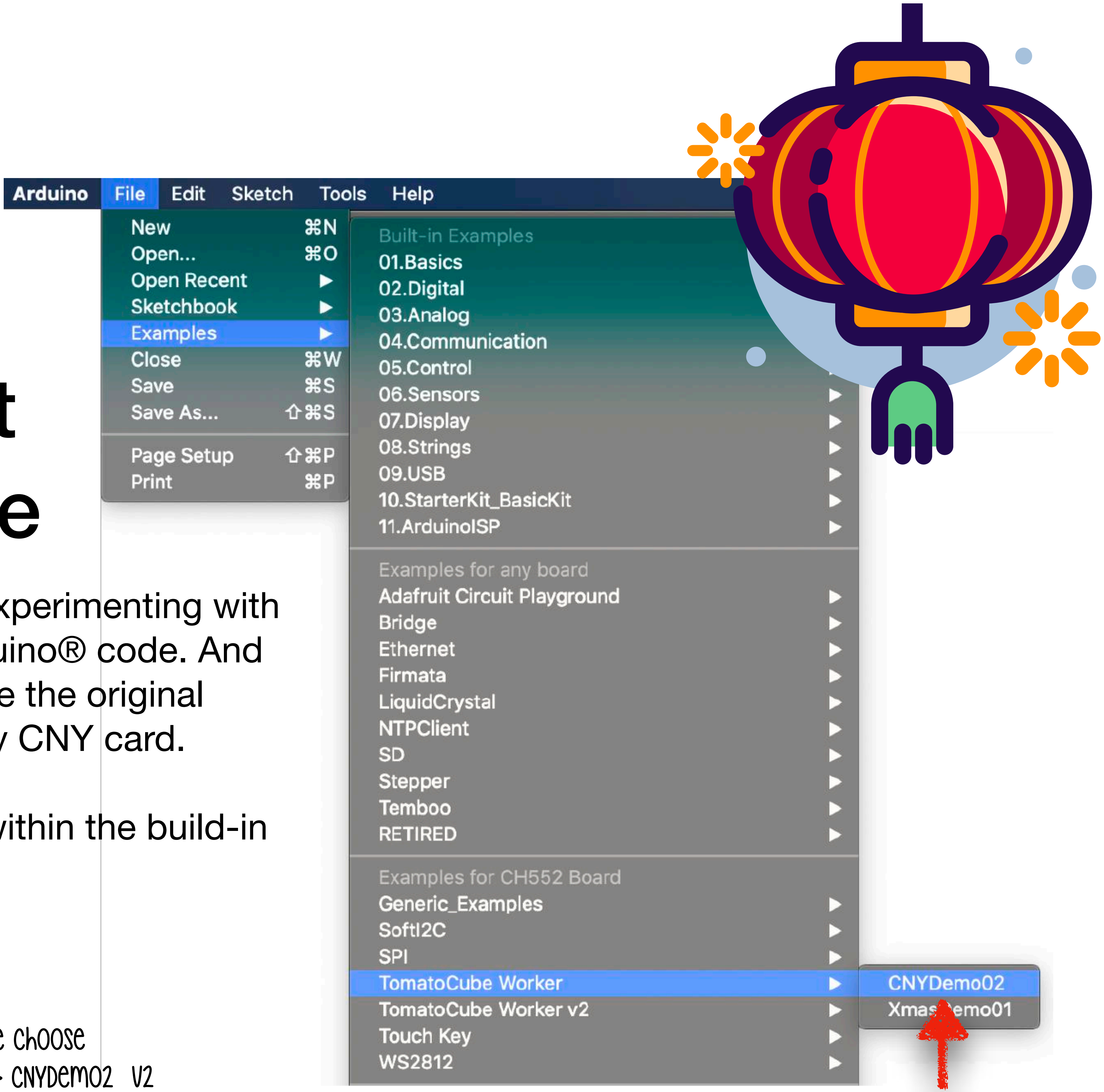
```
/*
 *  Ex_09 - MixingItup
 *  Combine LED, Touch & Tone
 *
 */

#include "TomatoCubeWorker.h"

unsigned char randomLEDSeq[10];
int currentLEDPattern = 0;
unsigned long lastLEDMillis = 0;
bool playMusicFlag = false;

const PROGMEM char cnySong[] = {
  NOTE_D4, 8, NOTE_E4, 8, NOTE_F4, 8, NOTE_G4, 8, NOTE_AS4, 4, NOTE_A4,  4,
  NOTE_A4, 8, NOTE_D5, 8, NOTE_D5, 8, NOTE_A4, 8, NOTE_A4, 4, NOTE_G4,  4,
  NOTE_G4, 8, NOTE_AS4, 8, NOTE_A4, 8, NOTE_G4, 8, NOTE_G4, 4, NOTE_F4,  4,
  NOTE_F4, 8, NOTE_E4, 8, NOTE_D4, 8, NOTE_CS4, 8, NOTE_D4, 4, NOTE_D4,  4,
  NOTE_G4, 136, NOTE_A4, 16, NOTE_F4, 136, NOTE_A4, 16, NOTE_E4, 8, NOTE_A4,  8,
  NOTE_D4, 8, NOTE_A4,  8, NOTE_G4, 136, NOTE_A4, 16, NOTE_F4, 136, NOTE_A4, 16,
  NOTE_E4, 8, NOTE_A4,  8, NOTE_D4, 4, NOTE_G4, 136, NOTE_A4, 16, NOTE_F4, 136,
  NOTE_A4, 16, NOTE_E4, 8, NOTE_A4,  8, NOTE_D4, 8, NOTE_A4,  8, NOTE_G4, 136,
  NOTE_A4, 16, NOTE_F4, 136, NOTE_A4, 16, NOTE_E4, 8, NOTE_A4,  8, NOTE_D4, 4
};

                                        ...
```



https://tomatocube.com/url/e9a8wm

Code X - Restoring the Default Demo code

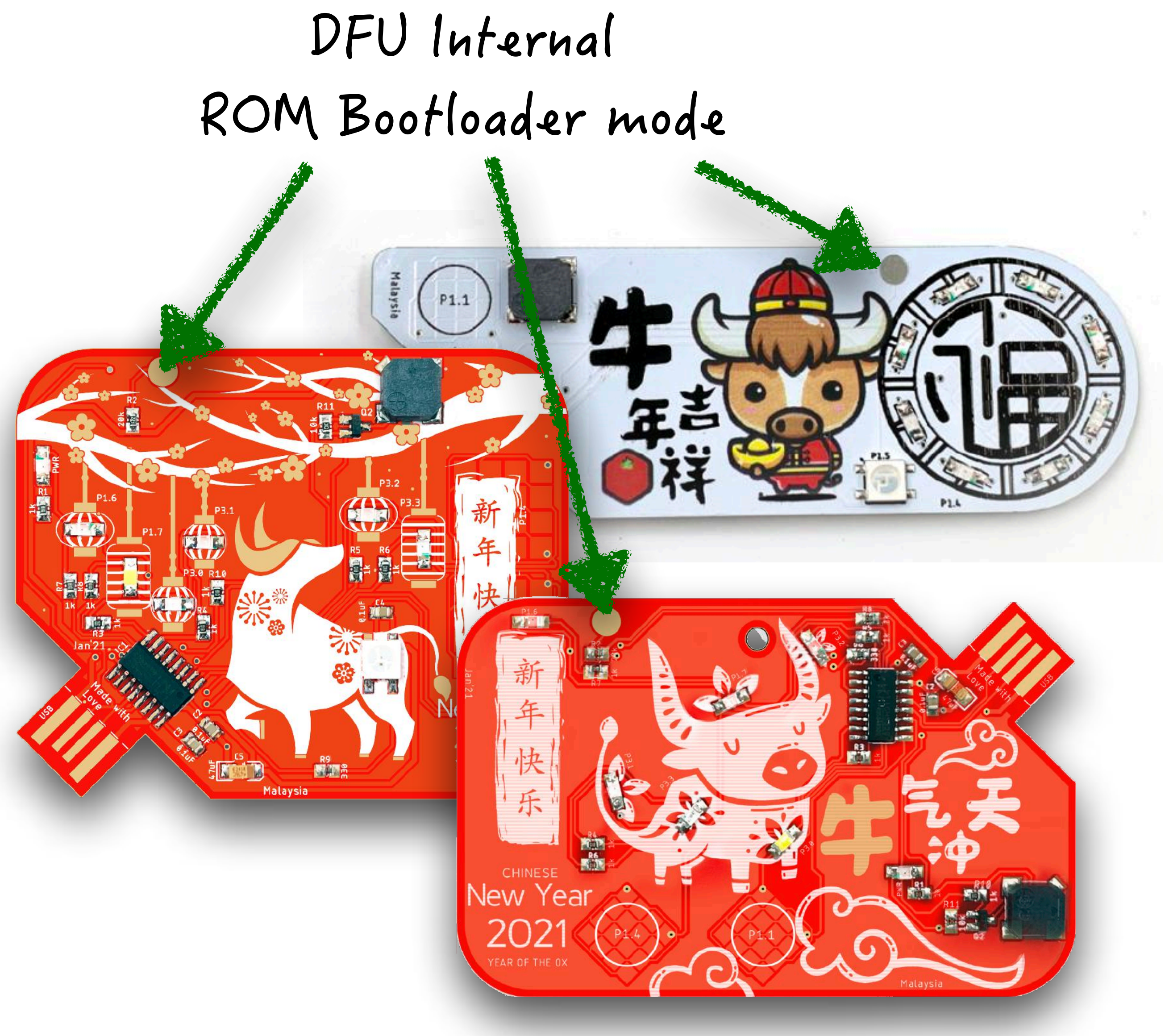
- Once you are done experimenting with writing your own Arduino® code. And you decides to restore the original demo onto the Happy CNY card.
- The Code is hidden within the build-in example codes.



For V2 Board, please choose
TomatoCube Worker V2 -> CNYDemo02_V2

Appendix A - Rescue a corrupted firmware

- There are various reasons why you will need to force the board to enter boot-loader mode.
  - A) **Corrupted** firmware **upload** or **bad firmware.**
  - B) Your firmware **repurpose** the CNY card into some fancy **USB device.**
- We have a "**crocodile-clip**" pads to allow our CNY cards to boot into the on-chip **ROM boot-loader**.
  - One will need to short (connect) the pad on the top with the one on the bottom surface.



Appendix A - Crocodile Clip

Using a proper crocodile clip will be the easiest.

Appendix A - Paper Clip Hack!

For the Macgyver among us, you can create a makeshift jumper using normal everyday metal paper clip.