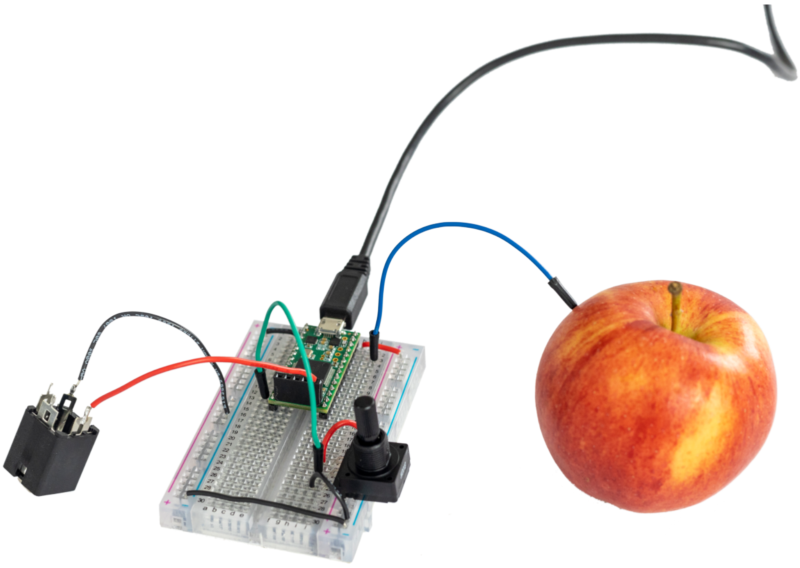
LABBOOK

TEENSYnth

DIY

3

PROGRAMMING AND SOUND



LABBOOK

TEENSYnth

Programming and sound.3

Author: Tara Pattenden

This lab book is a set of instructions that will guide you through building the TEENSYnth. The TEENSYnth lets you use fruit to play sound! It’s built using a Teensy, a microcontroller that can be programmed to do many things.

Levels of difficulty:

1. AN INQUIRING MIND
   Suitable for anyone
2. RESEARCHER
   I'm not familiar with the field, but I'll manage with a bit of effort
3. EXPERT
   I have enough knowledge to work independently
4. MASTER
   I'm quite skilled and possess an in-depth understanding of the tricks of the artistic and scientific trades
5. DEVELOPER
   I have sufficient knowledge to be capable of guiding those from categories 1–3
6. MENTOR
   I understand the content, I’ve mastered the technology, I develop new knowledge independently and pass it on to others

Index

TEENSYnth

This lab book is a set of instructions that will guide you through building the TEENSYnth. The TEENSYnth lets you use fruit to play sound! It's built using a Teensy which is a microcontroller that can be programmed to do many things. This book will include other projects and resources so you can continue building different things with the Teensy after the workshop is over. We will build one and explore the world of sound and sensors.

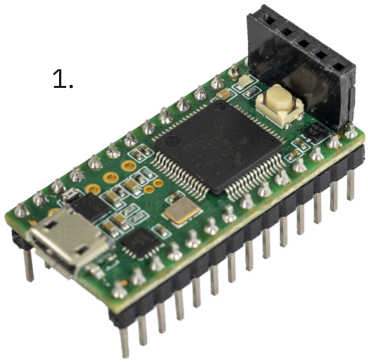
What do we need

1. Teensy 3.2 with pins
2. Breadboard
3. Jumper wires
4. Wire (cca 15-20 cm long)
5. Audio output
6. Headphones or speaker
7. USB data cable - micro USB to USB A
8. Computer with Arduino installed
9. A piece of fruit

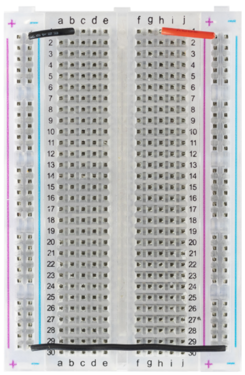
OPTIONAL:

10. 2 x 10K LIN potentiometer
11. LDR resistor

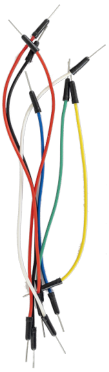
1.



2.

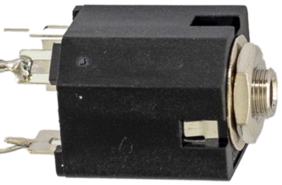


3.



4.

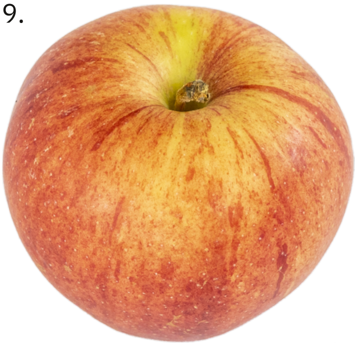
5.



6.

7.

9.



10.

11.

Set up the Arduino software

Let's start by setting up and testing the Arduino software on your computer.

Step 1 - install the required software

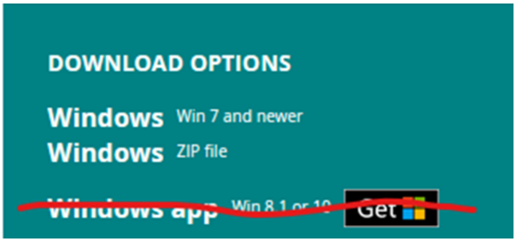
We communicate with the Teensy by using the Arduino software. You will need to install this to code for Teensy.

Download the following software:

Arduino IDE

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

If you are on a Windows machine - do not choose the app option as it will not allow you to install the Teensyduino software that is needed.



Teensyduino

https://www.pjrc.com/teensy/td_download.html

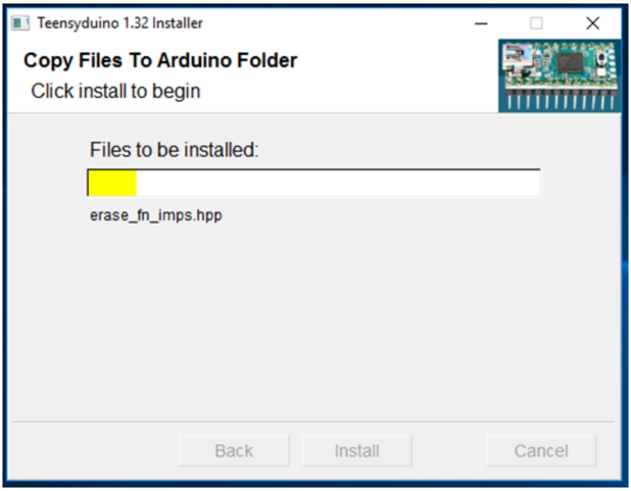
Install Arduino

Install and then run the Arduino software. Close the Arduino software.

Install Teensyduino

The Arduino software doesn’t have built-in support for the Teensy, so you must run the Teensyduino installer to add the Teensy files to your Arduino software.

The installer asks for the location of your Arduino software. Select the location where you installed Arduino. The "Next" button will only activate when a folder containing the Arduino software is selected.


Teensyduino 1.32 Installer

Copy Files To Arduino Folder

Click install to begin

Files to be installed:

erase_fn_imps.hpp

Keep clicking "Next" until the installation is completed.

For Windows Users

If you are using Windows, you should also run this Windows Serial Installer from https://www.pjrc.com/teensy/serial_install.exe. Choose the "USB Serial" option, this will allow the Windows Found New Hardware Wizard to properly find the driver for the Teensy.

Step 2 - Test the Arduino software

We will test that you have the software running correctly by loading a program onto the Teensy that will make its on board LED blink.

To do this open the file: *File > Examples > 01. Basics > Blink*

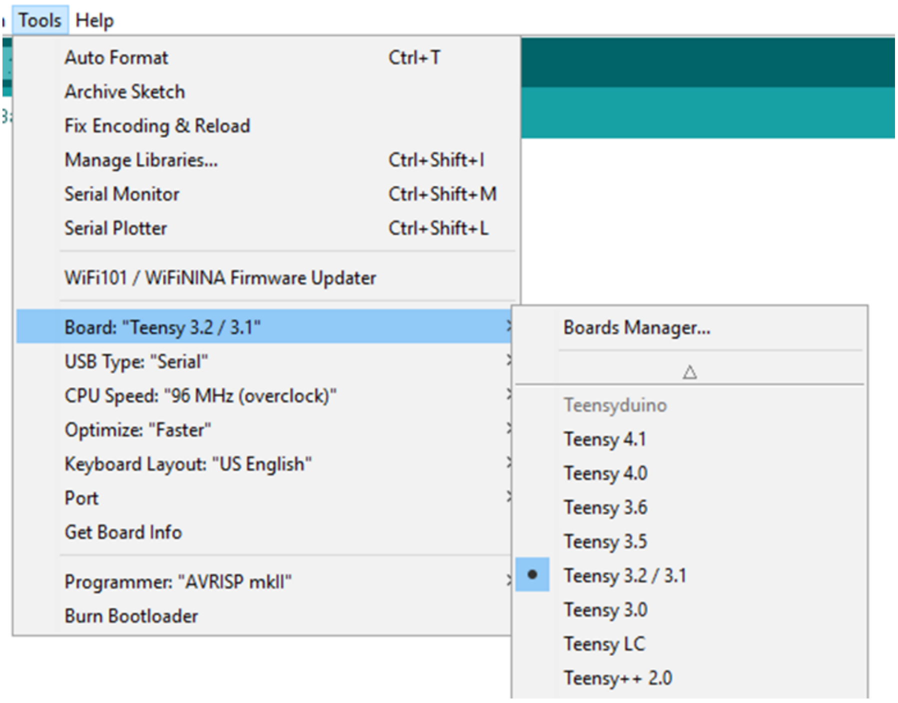
Now we need to send the program to our Teensy.

There are a couple of steps we need to set up first:

Set the board in Arduino

Check that the correct board is set in the program. This can be done in the tools menu

Tools > Board> Teensy 3.2/3.1

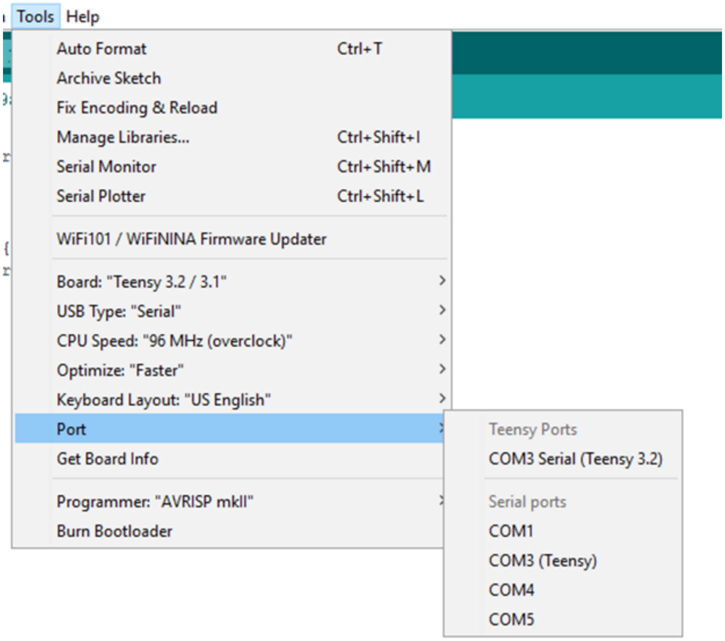

Tools
Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater

Board: "Teensy 3.2 / 3.1"
USB Type: "Serial"
CPU Speed: "96 MHz (overclock)"
Optimize: "Faster"
Keyboard Layout: "US English"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader

Boards Manager...
Teensyduino
Teensy 4.1
Teensy 4.0
Teensy 3.6
Teensy 3.5
● Teensy 3.2 / 3.1
Teensy 3.0
Teensy LC
Teensy++ 2.0

Select the port in Arduino

Under the tools menu *Tools* > *Port* > Select the port that says Teensy


Tools
Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Teensy 3.2 / 3.1"
USB Type: "Serial"
CPU Speed: "96 MHz (overclock)"
Optimize: "Faster"
Keyboard Layout: "US English"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Teensy Ports
COM3 Serial (Teensy 3.2)
Serial ports
COM1
COM3 (Teensy)
COM4
COM5

Step 3 - Compile and Upload

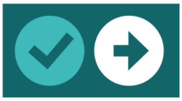
Now we will compile the code and upload it to the Teensy. Compiling the code is done by the software, it means that it is being made into a format that the Teensy can read.

Connect the Teensy to your computer with the USB cable.

In Arduino software, go to the menu *Tools* > *Boards* > *and choose Teensy 3.2.*

Then go to *Tools* > *Port* > and select the port that says Teensy.

Now click the upload arrow at the top right of the sketch:



The program will verify the code, compile it and then upload it to the Teensy board.

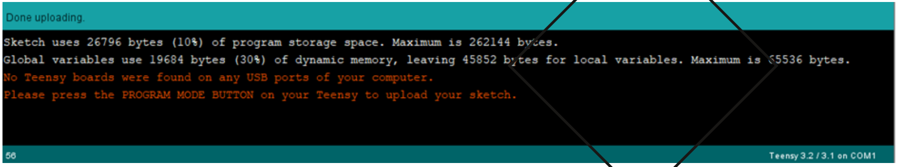
The tick button just compiles and verifies the code but does not upload it to a connected microcontroller.

Once the blink code is uploaded to your Teensy, the led on it should start blinking.

Common Arduino software issues

If your code does not upload there are several common problems what it could be. When there is an issue with the Arduino an error message shows at the bottom of the screen.

For example - in this case I don't have the Teensy connected to the computer.



```
Sketch uses 26796 bytes (10%) of program storage space. Maximum is 262144 bytes.
Global variables use 19684 bytes (30%) of dynamic memory, leaving 45852 bytes for local variables. Maximum is 65536 bytes.
No Teensy boards were found on any USB ports of your computer.
Please press the PROGRAM MODE BUTTON on your Teensy to upload your sketch.
```

Things to check if your program is not loading:

Are you using a data rated USB cable?

Have you selected the correct port?

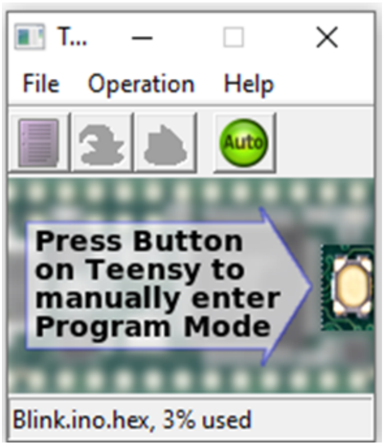
Have you selected the correct board?

Check to see if the Teensy loader is giving you an instruction.

https://coolcomponents.co.uk/blogs/news/common-arduino-issues

Teensy Loader

The Teensy Loader is how your Teensy board communicates with the Arduino software. It installs automatically with Teensyduino that you installed earlier. It runs in the background and generally you don't need to use it. However sometimes you may see an error message that you need to manually operate it - in these cases you press the button on the Teensy to enter manual mode and upload the program to it.



What is the Teensy?

Let's start by looking at the Teensy. Teensy is a type of microcontroller. A microcontroller is like a small computer that can run a program. It has 22 connection points that can be used as inputs and outputs - this is decided in the programming.

We will be detecting touch input from an apple and use that to play a sound.

There are many versions of the Teensy available. We will be using a Teensy 3.2 as it is suitable for our requirements.

9 of the connection points on the Teensy can be used for capacitive touch. This is what we will use to make a piece of fruit play a sound.

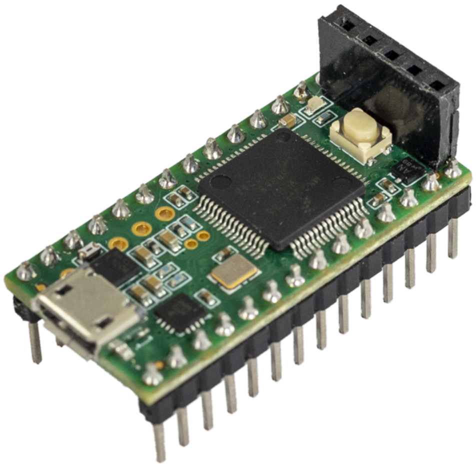
What is capacitive touch?

Capacitive touch is a proximity sensing technology. Capacitive sensors work by generating an electric field, and detecting nearby objects by sensing whether this field has been disrupted.

Capacitive sensing is all around us. It’s the same technology used by smartphone screens to detect touch.

Many everyday objects can be used as a capacitive touch sensor, they just need to be made of a material that is conductive - that is, it conducts electricity. Objects made from most metals are good conductors. Fruit and organic matter are also conductive and can be used as sensors. How cool is that!

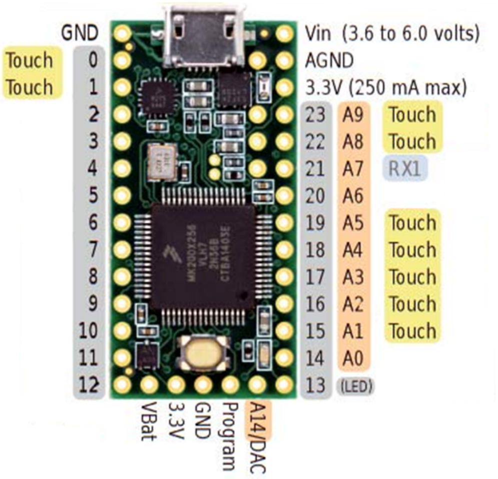
Let's take a look at the Teensy 3.2



Your Teensy will probably have terminal pins already attached and will look like this.

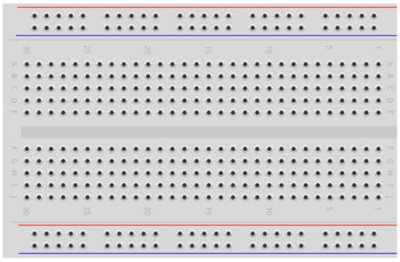
Each of these legs on the Teensy connects to an input/output. These are called pins. The pins can be programmed to do a variety of things. We are going to use them for capacitive touch.

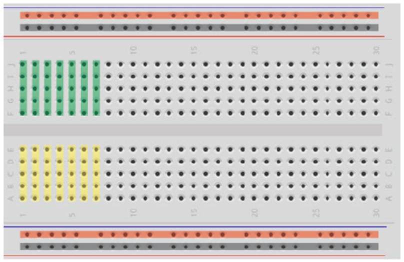
Pin 0, 1, 15, 16, 17, 18, 19, 22 and 23 can be used for capacitive touch.



Breadboard

This is a solderless breadboard. It allows you to hook up things to the Teensy without using any soldering.





Colored lines mark the holes that are connected inside the breadboard

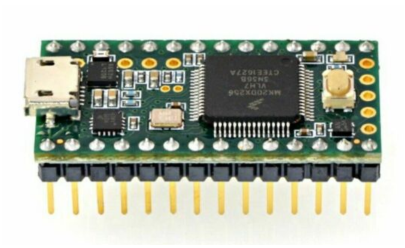
Inside the breadboard are bits of copper that connect the holes that are in the same row.

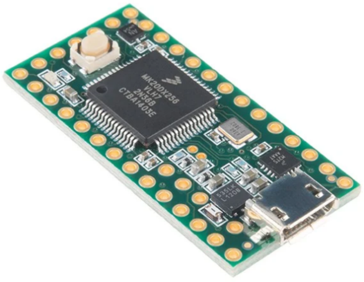
Preparing your Teensy to use with a breadboard

Depending on the parts you have, you many need to do a bit of soldering. There is a great comic called “Soldering is Easy” that you can download for free from the following location - https://mightyohm.com/files/soldercomic/FullSolderComic_EN.pdf

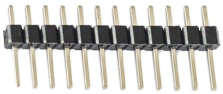
Teensy

The Teensy can come with or without pins. If it does not have pins you will need to solder them down both lengths of the Teensy so that you can use it with the breadboard.


Teensy 3.2 with pins

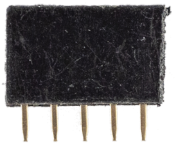

Teensy 3.2 without pins

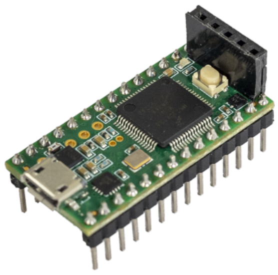
You can use a strip of male pin header like this:



Tip: place the pins in the breadboard and put the Teensy on top to hold them in place while you solder. Solder fairly quickly so that they don’t heat up too much and melt the breadboard.

To access the connections of the Teensy, we will solder 5 female pin headers on top of the board, so we can connect to these without having them connect to the breadboard. Your Teensy should come with soldered input connectors across this bottom row. If it does not you will need to solder this connection on yourself.





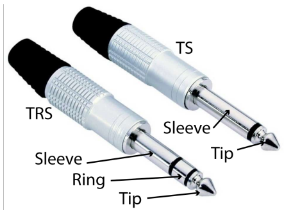
Audio

If your audio jack does not have any wires on it, you will need to solder wires to it. One leg of the audio input will be for ground, the other is for the audio output.

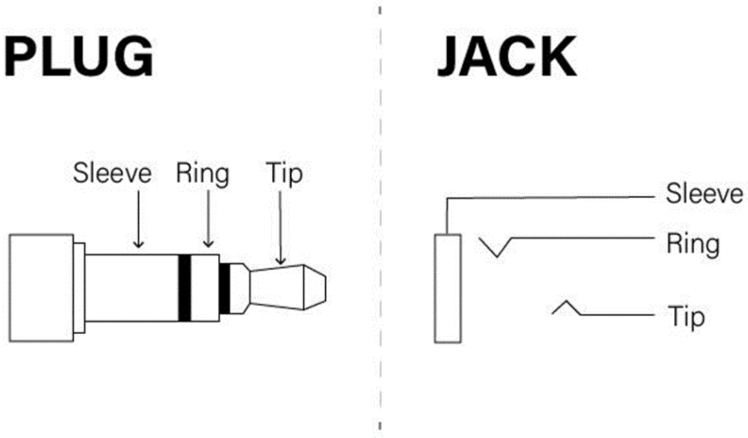
We will be using an audio jack to connect a speaker to the DAC. This connector is a stereo connector. The Teensy output is a mono connection. You can use a stereo or mono jack.

How do audio jacks work?

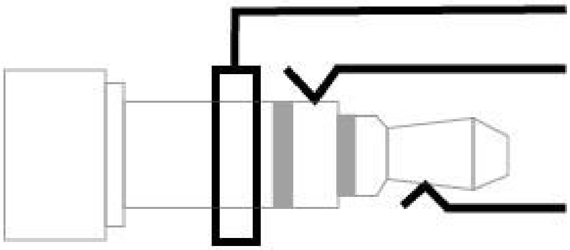
TS & TRS connectors are commonly used for audio. They come in different physical sizes, the most common are 3.5mm (minijack) and 6.35mm (guitar jack). TS is a mono connector and TRS is a stereo connector. They stand for: Tip Sleeve; and Tip Ring Sleeve.



This image is of 6.5mm TRS & TS plugs. The tip, ring and sleeve are all separate connections. The sleeve connects to ground and the tip and ring to the mono or stereo channels.



The plug and jack connect together like this:



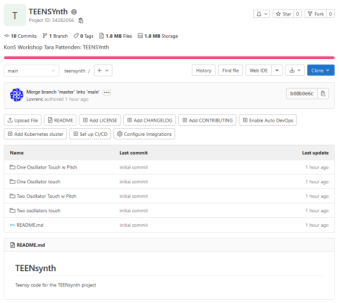
DAC Output on the Teensy

The audio output (DAC = Digital Analogue Converter) is on pin A14/DAC of the Teensy. It is at the bottom end of the Teensy, opposite side to the USB port.

Exercise 1 - Fruit sounds

Step 1

Download the project files. The files that you require can be found at https://github.com/problemmaths/TEENsynth



TEENSYnth
Project ID: 34282056

10 Commits · 1 Branch · 0 Tags · 1.8 MB Files · 1.8 MB Storage

KonS Workshop Tara Pattenden: TEENSYnth

main · teensynth / · History · Find file · Web IDE · Clone

Merge branch 'master' into 'main'
Lovrenc authored 1 hour ago · b88b0e6c

Upload File · README · Add LICENSE · Add CHANGELOG · Add CONTRIBUTING · Enable Auto DevOps · Add Kubernetes cluster · Set up CI/CD · Configure Integrations

Name	Last commit	Last update
One Oscillator Touch w Pitch	initial commit	1 hour ago
One Oscillator touch	initial commit	1 hour ago
Two Oscillator Touch w Pitch	initial commit	1 hour ago
Two oscillators touch	initial commit	1 hour ago
README.md	initial commit	1 hour ago

README.md

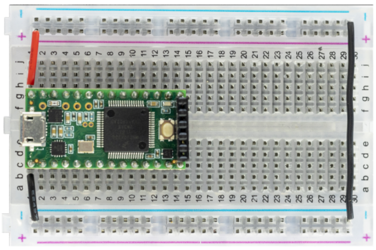
TEENsynth

Teensy code for the TEENsynth project

Click on the button that says code and select the "Download ZIP" option to download all the files. Unzip them to a folder on your computer and remember where you have saved them.

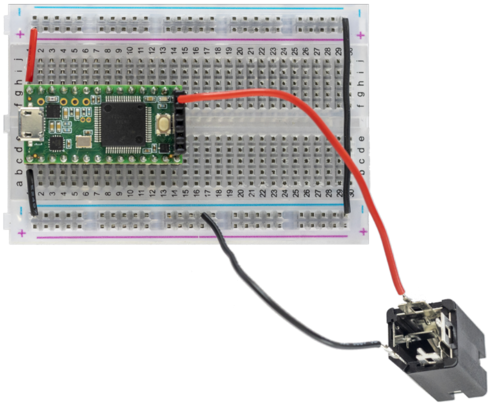
Step 2

Place the Teensy on the breadboard so that it sits across the middle ditch and press firmly until the pins are completely inside the board. Connect the Teensy VCC (+) and GND (-) to the breadboard. Some of the parts we are going to add will need to be connected to power and ground. We will take the power from the Teensy and connect it to the side rails so we can use them for positive (+) and negative (-) power.



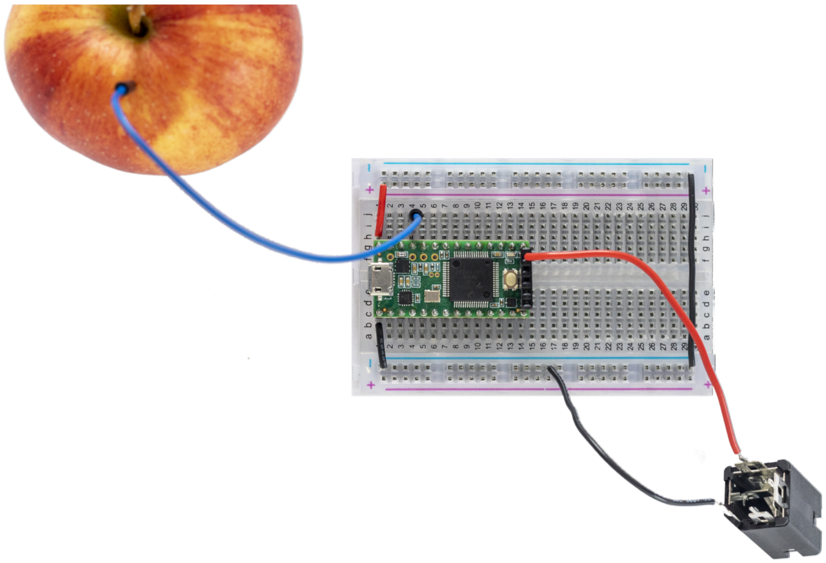
Step 3

Attach the red wire from the audio connector to the DAC - which is on the opposite side of the board to the USB input (check the chart on page 13 if you need a reference) and the black wire from the audio input to the GND on the breadboard.



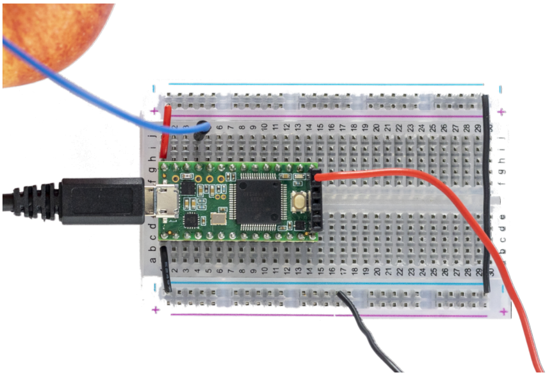
Step 4

Attach the apple to pin 23 (A9). Stick the end of the jumper wire straight into the apple!



Step 5

Plug in the USB cable to power the Teensy.

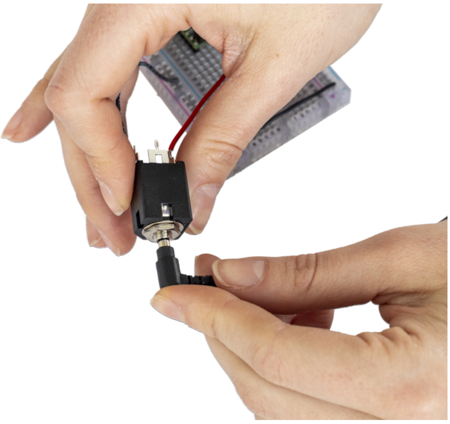


Step 6

Open the file called one_osc_touch.ino in the Arduino software. Compile and upload it to your Teensy by pressing the arrow symbol as you have done earlier.

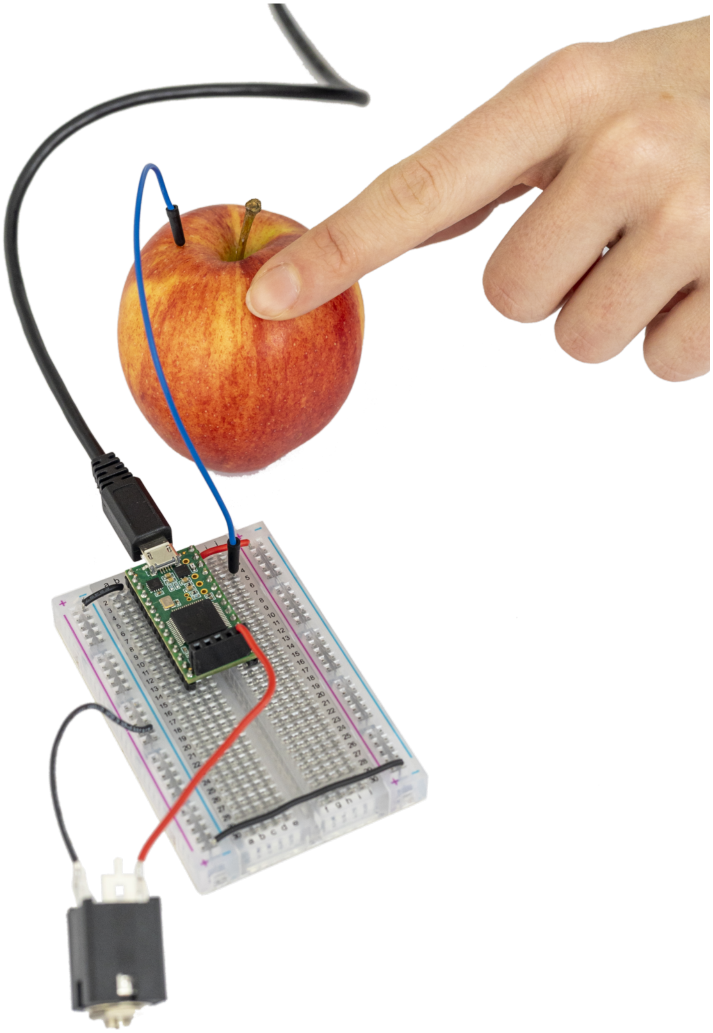
Step 7

Plug in your headphones to the audio output.



Step 8

Touch the apple, it should make a sound when you touch it. WOW!!!



Let's experiment with some different materials instead of the apple to see which materials are conductive - swap out the apple for something else and see if it's conductive. There is a list of materials you can try out in the appendix of this document.

Exercise 2 - Playing with pitch & Looking at Code

Now let's play around with the code to change the sound. We will explain what is happening in the code after this, for now we will just alter the relevant code.

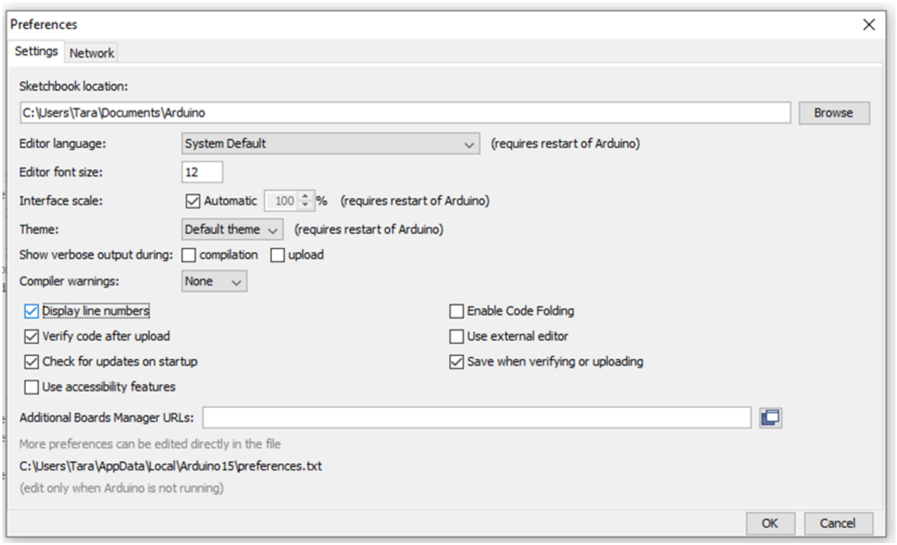
Step 1 - Open Arduino

Open the document called one_osc_touch.ino.

This is the program that you need to load onto the Teensy to make the piece of fruit play a sound.

Turn on Line numbers

We will be referring to the line number in the code later on. To see the line numbers in your Arduino code, go to the file *menu > preferences* (on Mac you need to go to Teensyduino in the top bar). Here you can check the box that says *Display line numbers*.



Preferences

Settings | Network

Sketchbook location:
C:\Users\Tara\Documents\Arduino [Browse]

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

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:

More preferences can be edited directly in the file
C:\Users\Tara\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

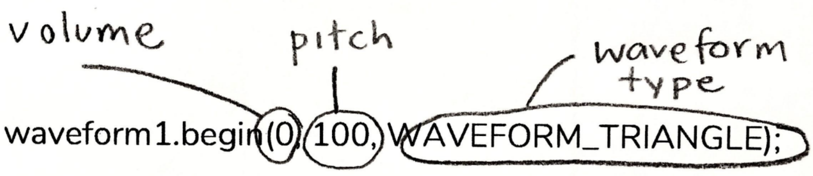
[OK] [Cancel]

Step 2 - Alter the code

Navigate to line 27 of the code.

It should say:

```
waveform1.begin(0, 100, WAVEFORM_TRIANGLE);
```



This line of code sets up the sound properties, that is the volume, pitch and waveform type.

We want to change the pitch. That is the second number in the brackets in Hz.

The human hearing range is commonly given as 20 to 20,000 Hz, although there is considerable variation between individuals. 20hz would be a very low pitch sound and 20,000hz would produce a very high pitch sound.

https://www.szynalski.com/tone-generator/

Let's change that second number - choose something between 20 – 20.000.

Step 3 - Compile & Upload

Now we will compile the code and upload it to the Teensy. Compiling the code is done by the software, it means that it is being made into a format that the Teensy can read.

Connect the Teensy to your computer with the USB cable.

In Arduino software, go to the menu *Tools > Boards > and choose Teensy 3.2.*

Then go to *Tools > Port > and select the port that says Teensy.*

Now click the upload arrow at the top right of the sketch.

The program will verify the code, compile it and then upload it to the Teensy board.

The tick button just compiles and verifies the code but does not upload it to a connected microcontroller.

Touch the apple and hear how the sound has changed.

Step 4 - Listen to different frequencies

Repeat steps 2 & 3, and try a few different frequencies. Here is a list of which frequencies correlate to notes on the musical scale.

https://pages.mtu.edu/~suits/notefreqs.html

Step 5 - Listen to different wave types

Now, let's listen to how the different waveforms sound. Change the type of waveform on line 28, to one from the list below, the waveform type is written straight after the frequency, it is currently set to WAVEFORM_TRIANGLE. Once you have changed the waveform type, compile and upload the code and listen to how the sound changes.

- WAVEFORM_SINE
- WAVEFORM_SAWTOOTH
- WAVEFORM_SAWTOOTH_REVERSE
- WAVEFORM_SQUARE
- WAVEFORM_TRIANGLE

Exercise 3 - Reading inputs

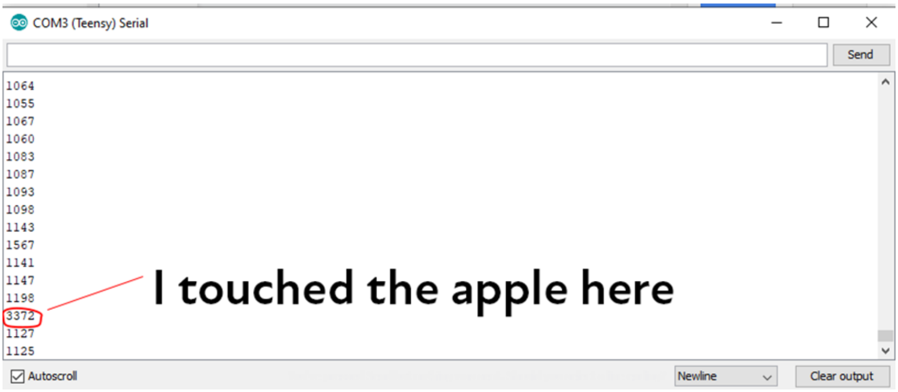
In this exercise we will use the Serial Monitor to see what data is being sent when we touch the apple.

Step 1 - Open the Serial Monitor

Making sure the Teensy is plugged in, open the Serial Monitor in Arduino software by clicking the magnifying glass icon in the top right corner of Arduino. This will open a window which will have numbers running down the screen. The Serial Monitor is a really useful tool that lets us observe what is happening with the Teensy.

Step 2 - Touch the apple

Touch the apple while watching the Serial Monitor. You will see the numbers change.



```
1064
1055
1067
1060
1083
1087
1093
1098
1143
1567
1141
1147
1198
3372
1127
1125
```

I touched the apple here

Step 3 - Change the sensitivity

If you need to make your apple input more or less sensitive to touch, you can do this by changing the threshold number that we use to tell if the apple is being touched. We may need to change this if the apple doesn't recognise our touch or recognises it before we have touched anything. This number should be higher than the numbers showing when you aren't touching the apple and lower than the numbers that show when you are touching it.

Line 19 of our program is where we set the threshold number.

```
int thresh = 2300; // variable to store the touch threshold
```

I found that 2300 works well for a variety of fruit. Including the apple.

Step 4 - Upload and test

Compile and upload your altered code and then play with the apple to see how it is affected.

Notes about the Serial Monitor

This following code defines what information is sent to the Serial Monitor.

In the setup - we connect to the Serial Monitor with the following code on line 33. If this code is not there the Serial Monitor will not connect to the Teensy.

```
Serial.begin(9600);
```

In the loop section of the code, we tell the Serial Monitor which information we would like to monitor. Using this code on line 50. In this case, current is a variable which holds the capacitive touch data.

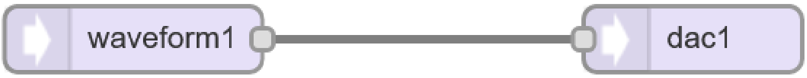
```
Serial.println(current);
```

Coding the Teensy

In this section we will look in depth at the code to play the apple using capacitive touch.

The Audio System Design Tool

Teensy tool for sound design is called The Audio System Design Tool. The audio tool can be used to set up the sound that you want to access with the code. It is a way to visually set up sounds, effects and mixers. It creates the code for you and is a great time saver. This is what the audio setup for the one fruit touch project looks like.



General info

The Teensy program is written in Arduino software in C++. Once you have written the code, it needs to be verified and compiled so that it can be sent to the Teensy. The verification process checks for any mistakes in the code - the code is case sensitive and punctuation marks are important.

We will start by using the code in the document. Don't worry if you don't know anything about coding. For now we can get started by running and altering the code provided. This is a good way to learn.

```
// Two slashes in front of the line mean that this is a comment and
// won't be read when compiling the program. You can use them to
// describe what is happening.
```

If you look at the code in Arduino, there are comments throughout that explain what the code does.

Let’s look at the code

Open the file one_osc_touch.ino in Arduino. This is the code that is currently on the Teensy that made the fruit play. Now let’s have a look at the code that makes this work. The concepts and code will be explained along the way.

```
//One oscillator capacitive touch
// using lots of code from Sebastian Tomczak
// https://little-scale.blogspot.com/2017/05/teensy-36-basics-
touchread.html
// this loads libraries that are required by Teensy
#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>
```

Loading libraries

This is the start of the document where we put information about what the code does and load the required libraries - the libraries are the items that end in *.h*. The code for the libraries is written for us using the audio editing tool, as you will see later.

```
// GUItool: begin automatically generated code
AudioSynthWaveform   waveform1;      //xy=387,283
AudioOutputAnalog    dac1;           //xy=549,283
AudioConnection      patchCord1(waveform1, dac1);
// GUItool: end automatically generated code
```

This code sets up the synthesizers we will be accessing - it is generated by the Audio Editing Tool.

What is a variable?

Here we set up the variables. Variables are virtual containers that store information under a name that we decide. To create a variable in Arduino, we declare the type of variable and the variable name and any value we want to assign to it - we don’t need to assign something immediately.

There are many reasons to use variables in coding. Some of these reasons are:

Efficiency - We use variables to save us time and so that we can easily make changes in the future. For example, we set up a variable called touchRead_pin, this variable stores the number of the input pin we will use to detect touch, it is referred to several times later in the code. If we want to change the number we can do it here in the variable. We could just use the pin number A9 throughout the code instead of a variable., but if we want to change it in the future we would have to look through the code and find everytime we used it and change it. It is much simpler to use a variable in this case

Storing dynamic data - Another reason to use a variable is to store data that is changing from an input.

There are different types of variables to store different types of data. We are using int and float.

Float - A float must be a number, it can contain decimals

Int - Integers are your primary data-type for number storage. They can only contain whole numbers, they can’t contain decimals

What is a function?

```
//set up variables
int touchRead_pin = 0; //variable to store the input pin
int thresh = 2300; // variable to store the touch threshold
int play_flag = 0; // variable that flags if sound is being played or not 1
= playing; 0 = not playing
int current; // variable used to store the value from the touch input pin
```

Void Setup()

This is necessary to run your code. The code in between the {} is run once at the start of your program running.

```
void setup() {
// put your setup code here, to run once:
```

AudioMemory

This is how you allocate audio memory to the Teensy to be used exclusively by the audio library. PUT AMOUNT!

```
AudioMemory(50); // Dynamic memory is allocated to be used
exclusively by the Audio library
```

Waveform.begin()

Setup the synthesizer sound with the parameters of volume, frequency and waveform type.

```
waveform1.begin(0, 100, WAVEFORM_TRIANGLE);
// this sets up the parameters of the wave form (volume, frequency,
waveform type)
```

Serial Monitor

You can use the Serial Monitor to see what data is being sent on the inputs. This code sets up the communication with the Serial Monitor, later we tell it what to monitor. 9600 is the speed at which the data is transmitted to the Serial Monitor. The data rate is measured in bits per second (baud).

```
// initialize serial communication with computer - sets up the serial
// monitor

 Serial.begin(9600);
```

Void loop()

The program runs through the items in the loop - the code within the {} then goes back to the start of the loop and does it again. This happens very very fast.

```
// This is the loop, the program runs through these items then goes
back to the start of the loop and does it again.
// This happens very very fast.
void loop() {
}
```

TouchRead()

The TouchRead function is used to read the value of the touch input. Touch read reads the electric field surrounding it and its value is changed when the sensor is touched. Look at the number as it changes, and take notice of what it is - a general guide for this threshold is 2300 if you need to change the sensitivity of the capacitive touch object, change this number in response to what you see in the Serial Monitor.

First we set the variable current to store the data coming from the capacitive touch from the apple.

```
current = touchRead(touchRead_pin); // setting the 'current' variable
// Now the value from the touchRead function is compared to the
threshold value.
```

We set up variables to store this information at the start of our program. The variables are current, thresh and play_flag. Our threshold variable - thresh - is set to 2300.

If()

An if statement checks to see if something particular is happening. If it is happening, the code within the {} is executed, if it isn't happening we move on to the next bit of code. We have two if statements here to check if the apple is being touched and if the sound is playing.

There are four different states our synth could be in with different actions required depending on what the state is.

State	**Indicated by**	**Outcome**
New Apple Touch	The touch value from the apple - current - is higher than the threshold and the sound is not playing.	Set the play flag to equal 1 to indicate playing. Turn the volume up.
Still touching apple	The touch value from the apple - current - is higher than the threshold and the sound is playing.	Do nothing.
Stopped touching apple	The touch value from the apple - current - is lower than the threshold and the sound is playing.	Set the play flag to 0 to indicate no sound. Turn the volume off.
Apple not being touched	The apple is not being touched - the touch value from the apple - current - is lower than the threshold and the sound is not playing indicated by the play flag being 0.	Do nothing.

```
 if(current > thresh && play_flag == 0) {
		play_flag = 1;
		waveform1.amplitude(0.2); // turn the volume up to 0.2  - 1 is
full volume.
  }
```

If the apple is being touched, the touch value - current - will be greater than the threshold. If it is also a new touch, i.e. your hand wasn't touching the apple already, the play_flag will be set to 0.

If the touch value - current - is above the threshold - thresh - and the note is currently not playing - play_flag is equal to 0 - then a note on event is generated and the play_flag variable is set to 1 indicating that there is now a note playing.

If the touch value - current - is above the threshold - thresh - and the note is playing - play_flag is equal to 1 -, no action is taken

Apple is not being touched, and the sound is playing.

```
if(current < thresh && play_flag == 1) {
        play_flag = 0;
        waveform1.amplitude(0);  // turn the volume to 0 - no sound
}
```

If the touch value is below the threshold and the note is currently playing, then a note off event is generated and the play flag is set to 0 indicating that there is not currently a note playing. If the touch value is below the threshold and the note is currently not playing, no action is taken.

```
Serial.println(current); // this sends the data from the input pin which
is stored in the variable -  "current" - to the serial monitor
```

Delay()

A delay measured in milliseconds. We have a short pause before the loop goes back to the start and replays.

```
delay(100);
```

AudioSynthWaveform

We refer to this as a variable in the code so that we can have multiple instances of it.

Look for waveform and waveform2.

Properties - waveform, amplitude, frequency

Waveform is the type of wave used. Supported waveforms are:

- WAVEFORM_SINE
- WAVEFORM_SAWTOOTH
- WAVEFORM_SAWTOOTH_REVERSE
- WAVEFORM_SQUARE
- WAVEFORM_TRIANGLE
- WAVEFORM_TRIANGLE_VARIABLE
- WAVEFORM_ARBITRARY
- WAVEFORM_PULSE
- WAVEFORM_SAMPLE_HOLD

Frequency

This controls the pitch of the sound. The human hearing range is commonly given as 20 to 20.000 Hz, although there is considerable variation between individuals. 20 Hz would be a very low pitch sound and 20.000 Hz would produce a very high pitch sound.

Amplitude

This controls the volume of the sound using a value between 0 - 1, where 0 is no volume and 1 is full volume.

Functions

You can set all the parameters of the waveform in one function like this:

```
variableName.begin(level, frequency, waveform);
```

In our code it looks like this, where waveform1 is the variable name for the waveform we refer to. This sets the volume to 0.5, the pitch (frequency) of the waveform to 300 Hz and the type of waveform to be a sine wave.

```
waveform1.begin(0.5, 300, WAVEFORM_SINE);
```

You can also set the waveform parameters individually like this:

begin(waveform);

Configure the waveform type to create.

Eg:

```
waveform1.begin(WAVEFORM_SINE);
```

frequency(freq);

Change the frequency.

Eg:

```
waveform1.frequency(300);
```

amplitude(level);

Change the amplitude. Set to 0 to turn the signal off.

Eg:

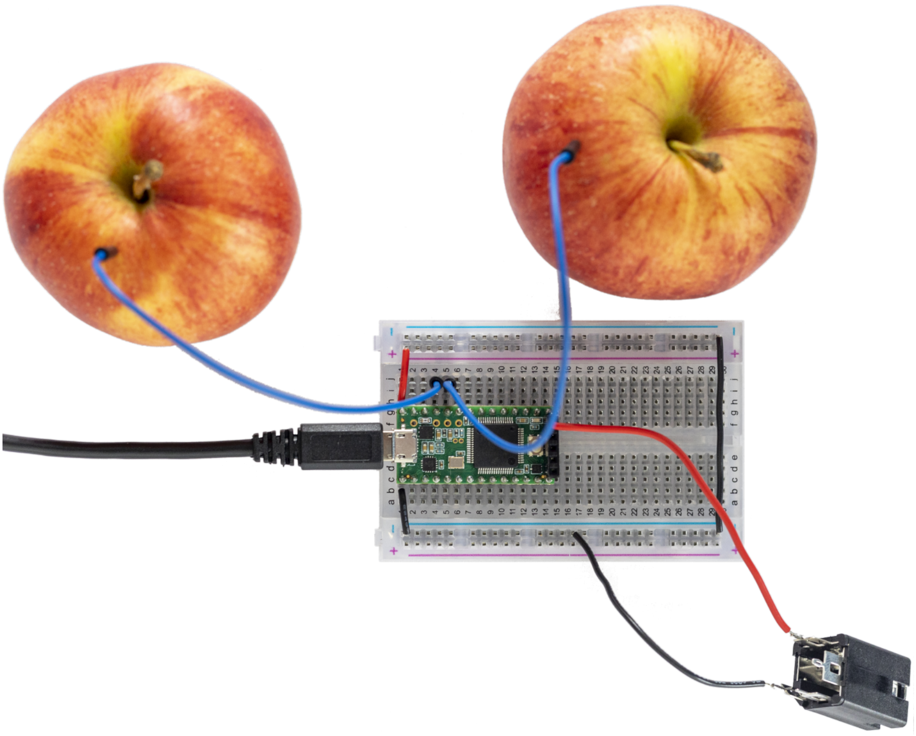
```
waveform1.amplitude(0.5);
```

Exercise 4 - Add more fruit

Now let's add a second apple to the Teensy so that we can have two notes.

Step One

Connect pin 22 (A8) of the Teensy to the second apple.



Step Two

Load the file two_oscillators_touch.ino.

Step Three

Upload the Arduino file two_oscillators_touch.ino to the Teensy and now listen to both apples.

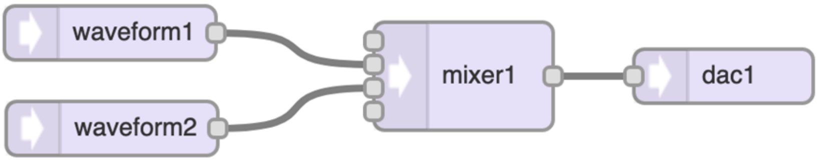
Let’s look at the code

These are the steps you need to follow to add the second apple to the Teensy. Have a look at these notes and then follow the same steps to add a third apple.

Step 1 - Update the Audio Design Tool

To add an extra apple into the code, we need to update the waveforms in the audio design system. And because we have more than one sound we also need to use a mixer.

Add in a mixer and an extra waveform. Connect the waverforms and the mixer as in this picture.



Select Export code and then paste it over the code at the start of the document up to where it says:

```
// GUItool: end automatically generated code
```

https://www.pjrc.com/teensy/gui/

Generated code

```
#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>

// GUItool: begin automatically generated code
AudioSynthWaveform    waveform1;      //xy=290,213
AudioSynthWaveform    waveform2;      //xy=291,266
AudioMixer4           mixer1;         //xy=487,237
AudioOutputAnalog     dac1;           //xy=632,237
AudioConnection       patchCord1(waveform1, 0, mixer1, 1);
AudioConnection       patchCord2(waveform2, 0, mixer1, 2);
AudioConnection       patchCord3(mixer1, dac1);
// GUItool: end automatically generated code
```

Step 2 - Add new variables

Duplicate the variables that we use to refer to the apple and its data.

First, we need to add a new variable so that we can tell the difference between the two apples. You will need to add a 2 at the end of the new variable name.

Add the following code on line 22 after the variable setup. This sets up new variables for the second piece of fruit that you will add.

```
//variables for the second piece of fruit
int touchRead_pin2 = A8; //variable to store the second input pin
int thresh2 = 2300; // variable to store the second touch threshold
int play_flag2 = 0; // variable that flags if sound from the second apple
is being played or not 1 = playing; 0 = not playing
int current2; // variable used to store the value from the second touch
input pin
```

At line 51 (approximately, yours may be slightly different), after where the current variable is set, paste this code:

```
current2 = touchRead(touchRead_pin2); // setting the 'current2'
variable
```

Step three - Set up the sound parameters

At line 37 after waveform1 parameters are set

```
waveform1.begin(0, 180, WAVEFORM_TRIANGLE); // this sets up the
parameters of the second wave form
```

Step Four - Check if the sound is playing

Check if the sound is playing when the apple is being touched. This changes the volume of the apple sound accordingly.

Add this code for the second apple checking if it is playing at line 60 after the first play flag is checked.

```
//repeat for the second apple
 if(current2 > thresh && play_flag2 == 0) {
          play_flag2 = 1;
          waveform2.amplitude(0.2); // turn the volume up to 0.2}
```

```
//repeat for the second apple
 if(current2 < thresh && play_flag2 == 1) {
          play_flag2 = 0;
          waveform2.amplitude(0);  // turn the volume to 0 - no sound}
```

Step 5 - Add more apples

That is all you need to do to add more apples. Now you try adding a third apple giving new variables a number 3 instead of 2

Exercise 5 - Controlling Pitch

In this exercise we will add pitch control to the apple.

Step One

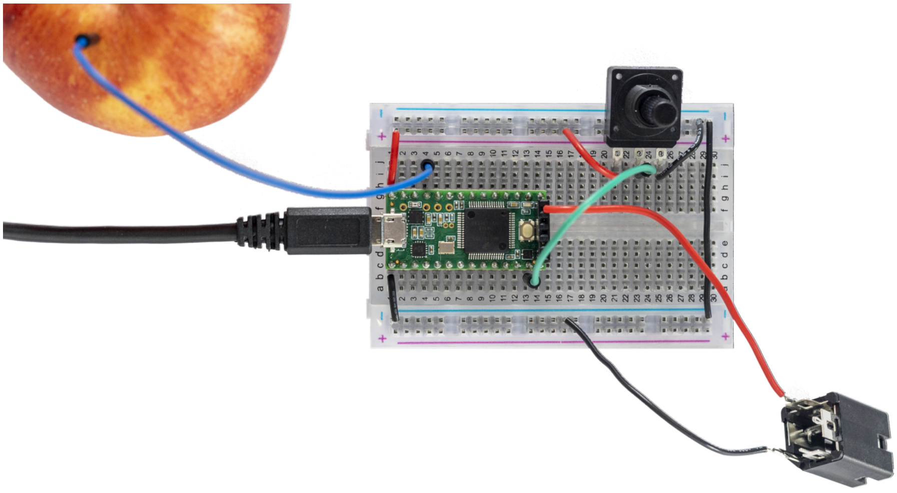
Remove the second apple from the breadboard.

Step Two

Add a potentiometer to the breadboard

- Connect its middle pin to pin 11 on the Teensy.
- Connect the left pin to the power.
- Connect the right pin to ground.

Note: if the pitch is moving in the wrong direction then try changing which side the ground and power are connected to on the potentiometer.



Step Three

Open the file *one_osc_pitch.ino* then compile and upload it to the Teensy. The potentiometer will now control the pitch while you touch the apple.

Let's look at the code that makes that happen, in particular the new elements of the code used to set up the potentiometer.

You can see on line 22 we set up the following variables:

```
int pitchPin = A0; // variable storing which pin the pitch potentiometer
will connect to
float pitchData; // variable used to store the data coming from the
pitch potentiometer
float scaledPitch; //variable used to store the scaled data from the
pitch potentiometer
```

As you can see from the comments in the code:

pitchPin is the variable that will store which pin we connect the middle pin of the potentiometer to**.**

pitchData is the variable which will store the values being sent from the potentiometer. Potentiometers will output a range from 1 - 1023.

scaledPitch is the variable which will be used to store the scaled value - this is explained in more detail below.

In the loop, on line 44 we set the variable pitchData to read the values coming in from the potentiometer.

```
pitchData = analogRead(pitchPin);
// Read the data coming from the pitch potentiometer and save it in
the variable pitchData
```

Map

Here we use the map function to map the range values output by the potentiometer to the desired range of pitch.

Potentiometers put out a value from 1 – 1023 as you turn it. This is the same for most standard potentiometers. We want to convert these numbers into an audio frequency range. To do this we use the map function to set our pitch range to be from 20 – 8.000 Hz.

The map function is really useful. It re-maps a number from one range to another. That is, a value of *fromLow* would get mapped to *toLow*, a value of *fromHigh* to *toHigh*, values *in-between* to values *in-between*, etc.

You write the function like this:

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

If you look in our code you can see that we are mapping the potentiometer range 1 – 1023 to an audible frequency range, in this case we are using 20 – 8.000 Hz.

```
scaledPitch = map(pitchData, 1, 1023, 20, 8000);
// Scale the value from the 'pitchData' variable and store it in the
'scaledPitch' variable
```

You can experiment with the numbers here to change the range of pitch in the potentiometer. Remember human hearing ranges from 20 to 20.000 Hz, however our hearing is most sensitive in the 2.000 – 5.000 Hz frequency range.

Controlling the pitch

Now we set the pitch of the waveform to be controlled by the potentiometer. The *scaledPitch* variable is the mapped potentiometer output.

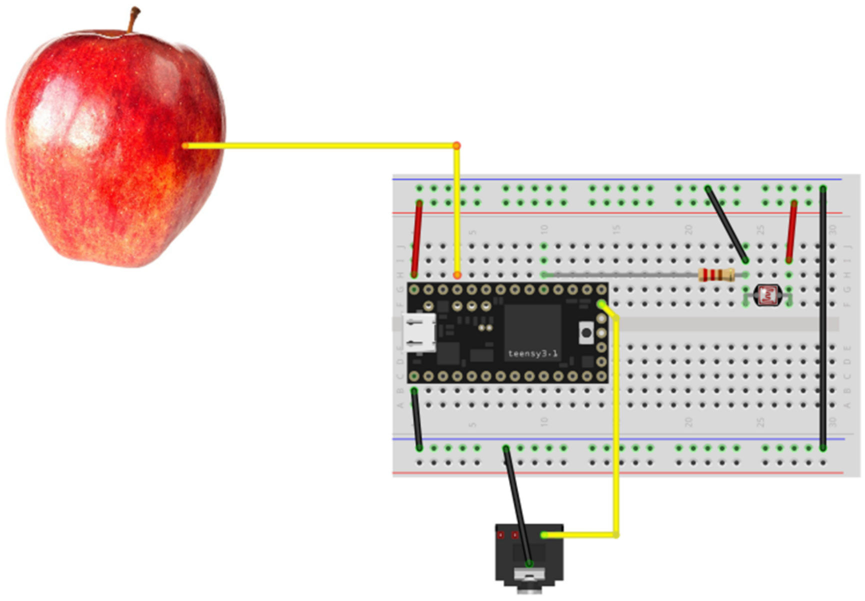
```
waveform1.frequency(scaledPitch); // Set the frequency of waveform1
to the value of the 'scaledPitch' variable.
```

Using an LDR

You could use an LDR (light dependent resistor) also known as a photoresistor instead of a potentiometer. As the LDR only has two legs and the potentiometer has three we need to wire it slightly differently.

One leg of the LDR needs to go to both ground - through a small resistor (3.3 – 10K) and the input pin A0 of the Teensy, the other leg goes to the positive power (+ v)

It would look something like this:



[Image source: Fritzing]

Appendices

Conductivity of Materials

Here are some examples of materials that are conductive and non-conductive.

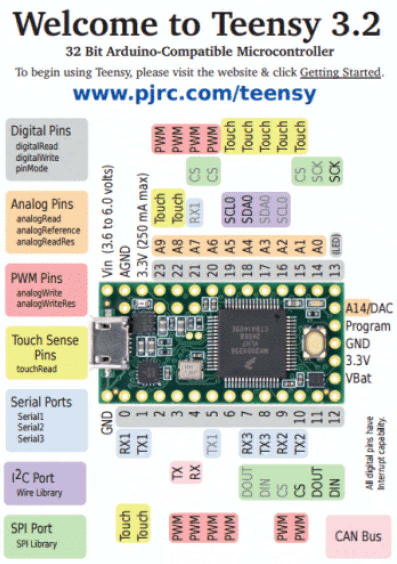
Conductive materials	**Non-conductive materials**
aluminium	teflon
platinum	glass
gold	rubber
silver	oil
ionised water	pure water
plants & fruit	fibreglass
iron	porcelain
steel	ceramic
brass	air
bronze	cotton
graphite	wood
	plastic

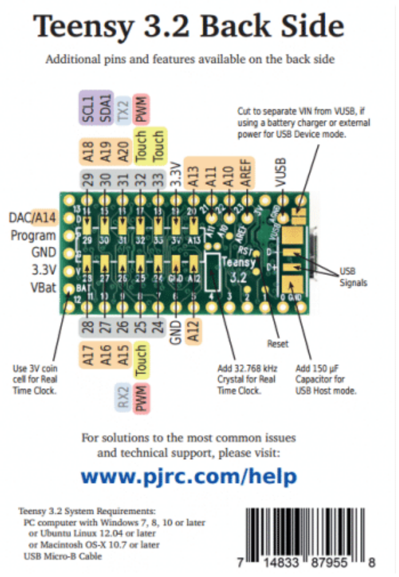
Breadboard

More information:

https://www.sciencebuddies.org/science-fair-projects/references/how-to-use-a-breadboard

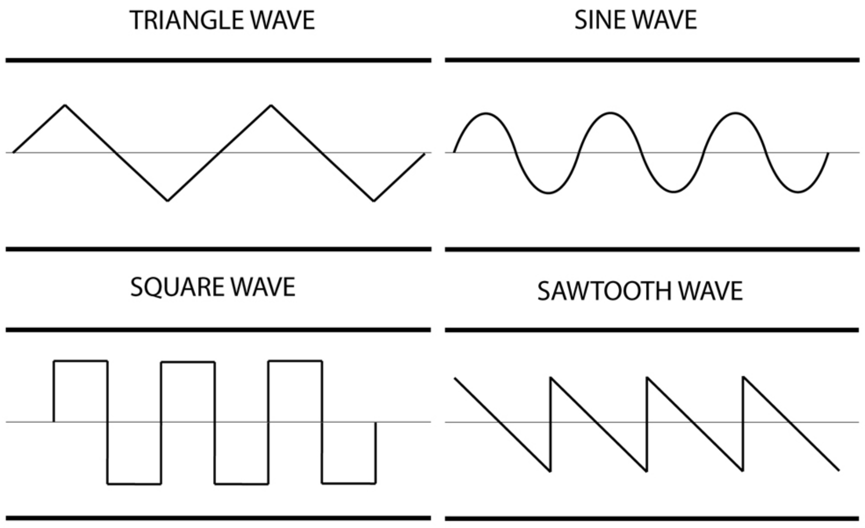
Teensy pinout




Teensy 3.2 Back Side
Additional pins and features available on the back side
Cut to separate VIN from VUSB, if using a battery charger or external power for USB Device mode.
DAC/A14
Program
GND
3.3V
VBat
USB Signals
Reset
Use 3V coin cell for Real Time Clock.
Add 32.768 kHz Crystal for Real Time Clock.
Add 150 µF Capacitor for USB Host mode.
For solutions to the most common issues and technical support, please visit:
www.pjrc.com/help
Teensy 3.2 System Requirements:
PC computer with Windows 7, 8, 10 or later
or Ubuntu Linux 12.04 or later
or Macintosh OS-X 10.7 or later
USB Micro-B Cable

Types of sound waves/What does sound look like?

Sound is made up of vibrations, or sound waves, that we can hear. These sound waves are formed by objects vibrating (shaking back and forth). The size and shape of sound waves determines the kind of sound heard. We can draw what the sound wave looks like.



Codes

https://gitlab.com/kons-platforma/teensynth

About the Author

Tara Pattenden is an artist, organiser and educator who works with a range of new media including electronics, sound, video and sculpture. Under the name Phantom Chips she uses the Teensy to make and perform with wearable, fabric-based electronic instruments that enable new expressive ways to play music. Her instruments invite audience participation by creating sound through movements and gestures (stretching, stroking and squeezing). She is based in Brisbane, Australia where she runs Elektrolab - a space for learning creative technologies. https://www.phantomchips.com/

Credits

Title
TEENSYnth, DIY

Labbook author
Tara Pattenden

Labbook editor
Tina Dolinšek

Technical editor
Lovrenc Košenina

Testing and review by
Tina Dolinšek, Lovrenc Košenina, Dunia Sahir, Uroš Veber and Rea Vogrinčič

Photos
Matjaž Rušt, Rea Vogrinčič, Fritzing

TEENSYnth Github
https://github.com/problemmaths/TEENsynth

Production
konS platform
Projekt Atol Institute

Front page banner
https://pixabay.com/photos/coding-computer-hacker-hacking-1841550/

Place and publisher
Novo mesto, LokalPatriot

Year
2022

Series
Labbook kons, 8th book

Free online publication

Notes

CIP - Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

004.4:681.5:534(035)
004.2/.3(035)

PATTENDEN, Tara
TEENSYnth : labbook : naredi sam (DIY) : programiranje in zvok 3 / [avtorica Tara Pattenden ; fotografije Matjaž Rušt ; prevod Zadruga Soglasnik]. - Novo mesto : LokalPatriot, 2022. - (Labbook kons ; knjižica 7)

ISBN 978-961-92137-6-6
COBISS.SI-ID 109368579

KONS.PLATFORM
FOR CONTEMPORARY
INVESTIGATIVE ART

konS = the Platform for Contemporary Investigative Arts is an open and evolving structure that seeks to establish links between communities, knowledge institutions, research centres and the economy at a systemic level, with all parties interested in co-creating a sustainable, safer and more ethical future in a dynamic and constantly changing world.

PARK

We create space in the hubs for young researchers and creative individuals and groups. We dedicate ourselves to exploratory and restless minds. With an inspiring program, we encourage the use of high technologies and at the same time cultivate critical thinking, encourage creativity and nurture innovation. Through active participation and capacity development, we create new creative communities. Our activities are intended for children, young people and the adult interested public.

kons-platforma.org

EVROPSKA UNIJA
EVROPSKI SKLAD ZA
REGIONALNI RAZVOJ

REPUBLIKA SLOVENIJA
MINISTRSTVO ZA KULTURO

Project konS – Platform for Contemporary Investigative Art is a project chosen based on the public call for the selection of the operations “Network of Investigative Art and Culture Centres”. The investment is co-financed by the Republic of Slovenia and by the European Regional Development Fund of the European Union.

Partners:

kersnikova, projekt atol, LJUDMILA, osmo/za, AKSIOMA, CONA, AKADEMIJA UMETNOSTI SCHOOL OF ARTS, MKC MARIBOR, MLADINSKI CENTER VELENJE, lokalPatriot

```
-indicator
nu .cart-icon
sparent heade
li.current_pa
li.current-me
 a:hover > .s
btn a:hover
li.current-me
lient-cart,.
olor:#fffff
op nav>ul>li
rea-toggle a
er transpare
```