Tiny trainable instruments

by

Aarón Montoya-Moraga

B.S., Pontificia Universidad Católica de Chile (2014)
M.P.S, New York University (2017)

Submitted to the Program in Media Arts and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2021

Author . . . .
Program in Media Arts and Sciences
August 20th 2021

Certified by . . . .
Tod Machover
Muriel R. Cooper Professor of Music and Media
Thesis Supervisor

Accepted by . . . .
Tod Machover
Academic Head, Program in Media Arts and Sciences

Tiny trainable instruments

by

Aarón Montoya-Moraga

Submitted to the Program in Media Arts and Sciences
on August 20th 2021, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

How can we build flexible multimedia instruments that we can train instead of program? How can we build and publish our own databases for artistic purposes? Tiny trainable instruments is a collection of instruments for media arts, using machine learning techniques and deployed in microcontrollers.

This thesis offers a proposal for the use of tiny machine learning for multimedia artistic purposes, with an emphasis on open source and machine learning ethics.

Thesis Supervisor: Tod Machover
Title: Muriel R. Cooper Professor of Music and Media

Acknowledgments

This thesis could not have been possible with the support of these wonderful people:

Aaron, Agnis, Aillaly, Alexandra, Allison, Alonso, Anamaría, Ana María, Andrea, Andrew, Baltazar, Beatriz, Belén, Benjamín, Bernardita, Braulio, Camila, Camilo, Carla, Carmelo, Carolina, Casey, Catalina, Charles, Christian, Claudia, Corbin, Cynthia, Daniel, Daniella, Dano, Devora, Dorothy, Erik, Euge, Felipe, Fernanda, Francesca, Francisco, Gabriela, Gaurav, Guillermo, Hannah, Jasmine, Javiera, Jen, Jennifer, Jorge, José, Juan, Julian, Juniper, Karina, Karsten, Kathy, Katya, Lauren, Lily, Lindsey, Lisa, Luna, Mahy, Manaswi, Marco, Margarita, María José, Martin, Maxwell, Mel*, Mitchel, Monica, Namira, Natalia, Natalie, Nathier, Newén, Nicolás, Nicole, Nikhil, Nina, Nouf, Nushin, Olivia, Pablo, Patricio, Pedro, Peter, Priscilla, Rafael, Raúl, Rebecca, Rébecca, Renata, Ricardo, Rodrigo, Rosalie, Roy, Russell, Ruta, Sankalp, Sasha, Sayén, Sean, Sebastián, Sejo, Shawn, Shir, Sofía, Sokio, Soledad, Tiri, TK, Tod, Tom, Verónica, Víctor, Victoria, Viniyata, Will, Wipawe, Yuan, Yining, Yuli, Yusuf, Zach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Me caí, me paré, caminé, me subí
me fui contra la corriente
y también me perdí

Soy Yo
Bomba Estéreo, 2015

1.1 Context

This thesis is the capstone project of my master's program, between the academic years 2019-2021 at MIT Media Lab, where I am a Research Assistant at the groups Opera of the Future and Future Sketches. The work presented here has been developed mostly while working remotely during the COVID19 pandemic, while at home in Boston MA.

This thesis is a collection of media arts instruments, using tiny machine learning, and with a strong emphasis on artificial intelligence ethics and Do-It-Yourself (DIY). Its main audience is beginners and artists, and it is my hope that this work can inspire a new generation of instrument makers, artists, designers, educators, programmers, policy makers, activists, and enthusiasts.

Machine learning has several barriers of entry, including cost, complexity, and difficulty. Since my practice relies on sharing, on open source strategies, and community building, I am not a fan of industrial machine learning that relies on proprietary software and hardware. This tends to happen because the ML models need to be trained for long periods of time, using expensive non open computational resources, with huge datasets that often are scraped from the internet without the explicit consent of the people, and a byproduct of surveillance capitalism.

Since machine learning algorithms are written by humans, and then trained on biased data, and they are deployed to the world, where they affect our lives. These imperfections are highlighted over the course of this thesis, by being explicit about the assumptions and quantizations performed when working with data.

This thesis proposes a scaling down of machine learning, to make it more accessible to artists, it is my belief that computational technology is

in expensive or rented resources, such as the service Google Colab, which allows you to share and train ML models on the cloud. This thesis shows users how to build machine learning art systems that can circumvent this and allow for more control of their data.

Artists have unprecedented access to new tools for making new tools and instruments, and this thesis intends to be a foundation for a new generation of instruments for manipulating audiovisual material, using machine learning. This approach is really exciting because it allows beginners and artists to train their instruments instead of programming them, by inputting data for tuning, instead of having to write lines of code and fixed thresholds for changing their behavior.

TODO: Add an example of a biased dataset that is commonly used.

The proliferation of surveillance tools and now of microcontrollers allows for an even more pervasive surveillance and data leaks by governments and corporations. This thesis allows for beginners and artists to develop their own databases, by self sensing and surveillance. TODO: explain self-sensing and surveillance, and explain harms of surveillance, with an example.

Low power and repurposable Tiny machine learning is defined as machine learning performed with low-power devices TODO: add citation or mention that it is my own definition. The proposal of media arts instruments that can run on little power, rechargeable batteries is a friendly use of resources for experimentation in arts. This is in contrast with the high power use and critique of other emerging fields such as Non Fungible Tokens (NFTs) and cryptoart. The proposal of these scriptable open source instruments allows for users to continuously tweak and modify their instruments, repurpose the hardware, and enable sharing.

TODO: Add a reference to an early tiny ML paper.

1.2 Objectives

With the release of TensorFlow Lite Micro, the TinyML Foundation, new avenues have been opened for creative expression using machine learning in microcontrollers.

TensorFlow is a library by Google for machine learning. They have released different flavors for different devices, including TensorFlow.js for working with KavaScript and in the browser, and TensorFlow Lite for devices with less memory, like mobile devices. The newer TensorFlow Lite Micro is even lighter and intended for working on microcontrollers, and that is the flavor I am using in this thesis.

in a paragraph here that combines the points you make in the first paragraphs -> the main contributions of YOUR thesis... which it seems is basically just addressing a bunch of the context section? Blakeley Payne's thesis does a good job of outlining contributions. It's also good to do here to understand where you are going

Homebrew examples, teach people how to build their own databases, and how to circumvent corporations view,

TODO: Terms and conditions comic book. If people were to read terms and conditions, it would take them X years

You can train an instrument to only detect your voice, your accent, your living conditions.

TODO: add blurb main inspiration about trail building and skatepark, and playpens, sandbox

TODO: add Marina Berry's paper on playgrounds, child development

TODO: Technology and Public Art with Rafael Lozano-Hemmer, June 3, 2020

`https://www.youtube.com/watch?v=QgVdEmqmuEE` 36m28s Rafael Lozano-Hemmer: Face recognition needs to be banned in all applications except art.

TODO: add Zoom example of garbage speech to text

TODO: Add contextual stories to frame each aspect TODO: Raspberry Pi example, it's cheap but it needs a lot of extra hardware to be used TODO: Also add examples about April Fools Day and Terms and Conditions, about owning your soul TODO: link https://www.youtube.com/watch?v=jOQ-9S3lOnM&t=125s TODO: Terms and conditions citation: (copied this in from my thesis, but check out McDonald and Cranor 2008) https://kb.osu.edu/handle/1811/72839 "Just considering privacy policies shown to people online in one year, it's been estimated that consumers would need approximately 244 hours to read, not skim, privacy policies shown to them in one calendar year."

1.3 Thesis outline

This thesis has cover the following chapters:

1. Chapter 1, Introduction: the context and summary of this thesis.
2. Chapter 2, Tiny trainable instruments: description of the design strategies for the software and hardware, description of the support team working on this thesis.
3. Chapter 3, Early experiments: my earlier work that led to this thesis, in the topics of media arts education, microcontrollers, and machine learning, among others.
4. Chapter 4, Background and inspiration: work by other people which has informed my work.
5. Chapter 5, Project evaluation: user feedback, field notes.
6. Chapter 6, Conclusions and future work: next iterations of the instruments, and their proposed use for educators and artists.

Chapter 2

Tiny trainable instruments

All the modern things
have always existed
they've just been waiting

The Modern Things
Björk, 1995

This chapter describes the process of conceptualizing and developing this thesis project, which is an antidisciplinary project, drawing from the disciplines of artificial intelligence, electrical engineering, computer science, media arts, music, and pedagogy, among others.

First there is a definition of the words that make up the title of this project, and then an overview and explanation of the main components of this thesis , which include:

1. TinyTrainable Arduino software library
2. Auxiliary code for machine learning
3. Workshop educational material

2.1 Definition of Tiny trainable instruments

2.1.1 Definition of tiny

Bi tiny I mean handheld, lightweight instruments that you can take for a walk or on your backpack. Tiny is also a nod to the new uprising field of tiny machine learning.

2.1.2 Definition of trainable

By trainable I mean a computational device that can learn by examples. This name comes from the machine learning lexicon, where training is a process where an algorithm finds the numerical values of all parameters (weights, biases) of a machine learning model, with the help of a dataset.

The process of training a model with a microcontroller has been really rewarding for me during the making of this thesis. When I program audiovisual computational devices to react to sensors, I often find myself having to look at streams of data, having to program statistical methods to detect trends, such as averages, and then having to program a hard coded value on the software, and having to write a manual for setup or fine tuning for collaborators to be able to run it again under different conditions.

The fact that I can program once, and then train on device, with the algorithm taking care of finding correlations, make for a more flexible workflow that I am hoping can be included in the toolkit of artists.

2.1.3 Definition of instruments

By instruments I mean devices that act as transducers across mediums, processing an input, which results in an output.

A musical example is the guitar, where the input strumming results in output sound. I mention the guitar because of its nonlinear and flexible behavior, with multiple different ways of strumming, even by non-human agents, resulting in infinite sounds.

Another of my favorite instruments is the bicycle, which I like to define as a device that converts the input pedalling into all sorts of outputs: adventure, sweat, and wind in your face.

2.2 Dreams and goals

We are surrounded by devices that are listening us.

TODO: insert picture of surveillance camera near my home, in the middle of nature.

I not only have algorithms

TODO: insert picture of garbled zoom speech to text, failing with my speech.

Dream: drum machine i can talk to, like The White Stripes also perform changes in realtime midset with no playlist.

The Tiny trainable instruments

Input \ Output	Buzzer	LED	MIDI	Printer	Screen	Serial	Servo
Color							
Gesture							
Speech							

Table 2.1: Matrix of inputs and outputs

Input	**Sensor library**	**Machine learning library**
Color	Arduino_APDS9960	Arduino_KNN
Gesture	Arduino_LSM9DS1	Arduino_TensorFlowLite
Speech	PDM	Arduino_TensorFlowLite

Table 2.2: Software dependencies for inputs

2.3 TinyTrainable Arduino software library

The main contribution of this thesis is the TinyTrainable Arduino library, an open source library for creating Tiny Trainable instruments, as in microcontroller-based devices that you can train to react to gestures with machine learning, so they can process and output different multimedia events, such as sound, movement, light, and text. s

2.3.1 Repository structure

The library is a repository hosted at `https://github.com/montoyamoraga/TinyTrainable`, where you can review all the history and commits through time, to see how the library or each file has evolved over time.

The structure of the folders follows 2 simultaneous specifications: it includes the necessary file and folder names for being packaged and indexed as an Arduino Library, and also it complies with GitHub guidelines for licensing and automatic workflows for testing the code.

The source code of the library is written in C++ and is located in the src/ folder.

Output	Actuator library
Buzzer	-
LED	-
MIDI	-
MIDI	-
Printer	Adafruit Thermal Printer Library
Screen	Adafruit_SSD1306
Serial	-
Servo	Servo

Table 2.3: Software dependencies for outputs

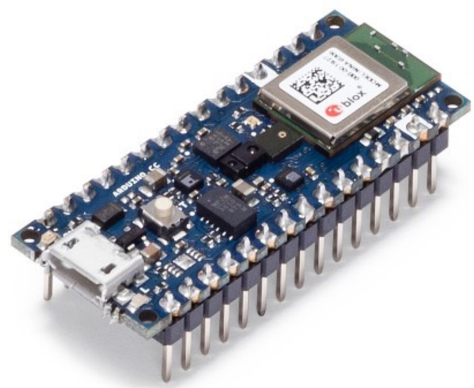


Figure 2-1: Arduino Nano 33 BLE Sense microcontroller with headers

Retrieved from [11]

The examples live in the examples/ folder, and are written in Arduino. Some trained machine learning models are included as C++ files on the assets/ folder.

2.3.2 Installation

The library can be downloaded from the Releases section of the repository at `https://github.com/montoyamoraga/TinyTrainable/releases`, where you also have access to the complete history of releases over time. To install it, you need to uncompress the .zip into a folder, and then make it discoverable by the Arduino IDE.

Since that method can be cumbersome and prone to errors, I made the effort to publish the TinyTrainable library, by complying with the latest Arduino Library Manager specifications, detailed on their repository `https://github.com/arduino/library-registry/`. With that, from the Arduino IDE you can open their Library Manager and do a one-click installation of the library, or even with the Arduino CLI you can install it via the command line on your machine.

2.3.3 Hardware basics

This library has only 1 fixed hardware requirement: it only runs on the Arduino Nano 33 BLE Sense, a microcontroller released in 2019. This library relies on the microcontroller's embedded sensors, so there is no need for wiring extra components.

For power you need a generic micro USB cable to provide the necessary input of 5V to the Arduino microcontroller. To upload code to the microcontroller you can use the same USB cable to connect to a computer running the Arduino IDE.

Figure 2-2: Micro USB cable

Retrieved from [10]

To build your instrument, you need a breadboard, jumper wires, and one of the many possible output devices that we describe in the following section.

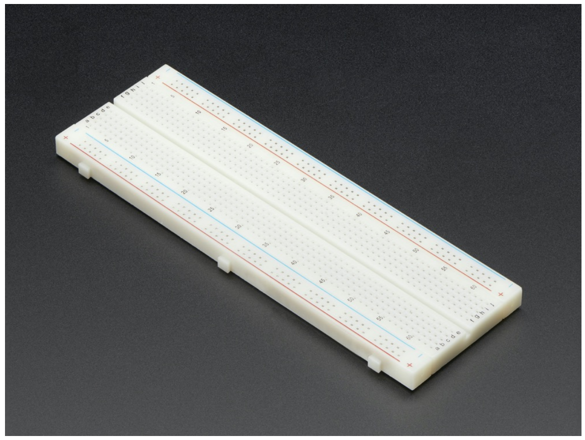


Figure 2-3: Breadboard

Retrieved from [3]

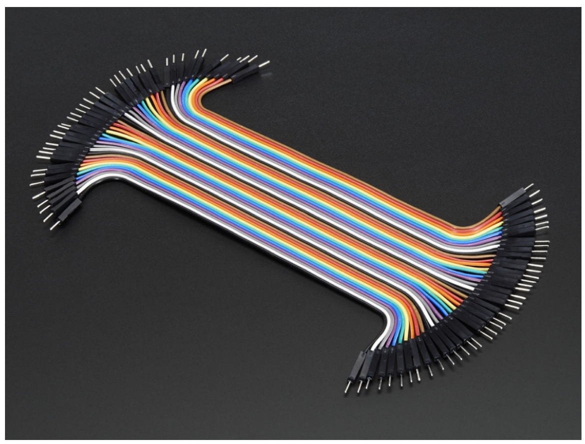


Figure 2-4: Jumper wires

Retrieved from [8]

2.3.4 Hardware for outputs

The TinyTrainable library supports a

UNTILHERE

2.4 Technology stack

This project is built with microcontrollers and

2.5 Design principles

This thesis tries to

1. Affordable
2. Hackable
3. Open
4. Private

2.5.1 Cheap

The materials for this thesis

2.6 Open

All examples included with this library were written with the aim of showing the fundamentals of how to build the instruments and different machine learning enabled manipulation of multimedia material, so that people could build on top of it and make it their own, by changing the values of variables and adding more functionalities.

2.7 Philosophy and experience

Throughout this project, the magic number was 3. The machine learning algorithms were hardcoded to be able to distinguish between 3 different categories: 3 colors, 3 physical gestures, 3 sound utterances.

2.8 Inputs

We are using the RGB color, proximity, gyroscope, accelerometer, and microphone sensors on the microcontroller, in order to capture the Inputs

1. Color
2. Gesture
3. Speech

2.8.1 Color

This approach uses the RGB color sensor from the microcontroller, with the auxiliary help from the proximity sensor, that is used to capture color information at a certain distance threshold.

The data is passed to a k-Nearest-Neighbor algorithm, programmed using the Arduino KNN library.

2.8.2 Gesture

This input uses the information from the Inertial Measurement Unit (IMU) of the microcontroller, including a gyroscope and accelerometer. It captures data after a certain threshold of movement is detected.

The data is passed to a TensorFlow neural network, programmed using the Arduino TensorFlow Lite library, and based on the included magic_wand example.

2.8.3 Speech

This input uses the information from the microphone of the microcontroller.

The data is passed to a TensorFlow neural network, programmed using the Arduino TensorFlow Lite library, and based on the included micro_speech example.

2.9 Outputs

The different outputs were picked, because of their low cost, ubiquity, and possibilities of expansion and combining them.

2.9.1 Buzzer

This output creates pitched sound, by using a PWM output.

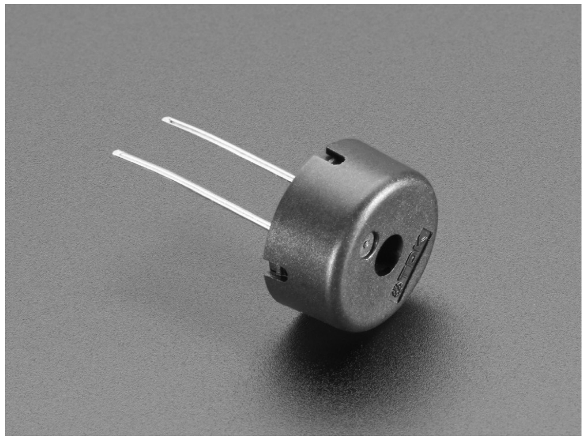


Figure 2-5: Buzzer

Retrieved from [7]

2.9.2 LED

This section requires no dependencies.

Figure 2-6: LED

Retrieved from [9]

2.9.3 MIDI

We wrote functionalities to manipulate MIDI instruments, and included examples to interface with some popular and cheap MIDI instruments, such as the Korg volca beats.

We included examples for rhythmic and melodic elements, using two very ubiquitous and inexpensive MIDI musical instruments, which are the Korg volca beats, and the Korg volca keys.

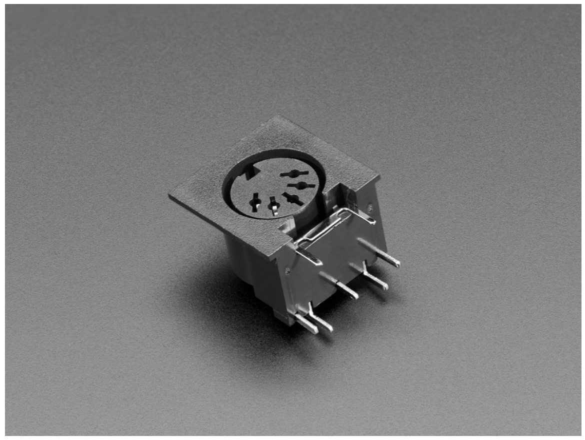


Figure 2-7: MIDI jack

Retrieved from [2]

2.9.4 Serial

This output requires no library dependencies.

We use the already mentioned USB cable to connect to our computer, and receive messages over the serial port, available through the Arduino IDE.

2.9.5 Screen

This output requires a library for printing messages on a screen.

2.9.6 Servo motor

This output creates movement and through that, rhythmic sounds.

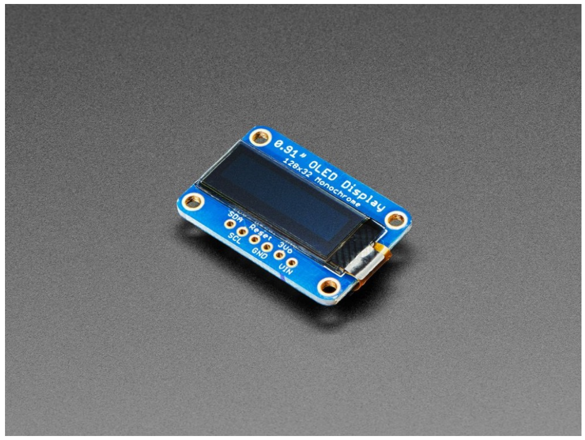


Figure 2-8: Screen

Retrieved from [6]

The main inspiration for this output was the emerging use of motor-activated percussive instruments, such as the Polyend Perc.

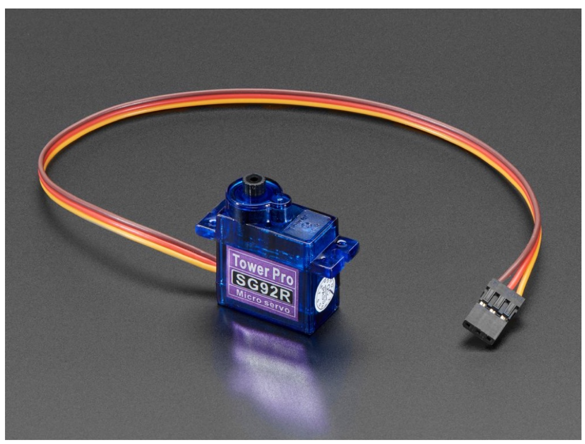


Figure 2-9: Micro servo motor

Retrieved from [4]

2.9.7 Thermal printer

A thermal printer is the basis for creating written and literary output, inspired by the field of computational poetry.

I used the popular Adafruit Thermal printer kit, which is documented on their website and includes a software library, distributed over GitHub and Arduino IDE, and also as a submodule on this project's TinyTrainable software library.

Figure 2-10: Thermal printer kit

Retrieved from [5]

2.10 Development

This thesis has been developed with the invaluable help of undergrad researchers Peter Tone and Maxwell Wang.

They have cloned both repositories, the main one and the Arduino library one, and have continuously submitted pull requests with their contributions.

Peter Tone has helped with research in data structures, library writing, and we have shared back and forth code, going from experimental proofs of concepts, and has also helped with the design of the user-facing library.

Maxwell Wang has proofread the code, has run the examples, and has helped with the writing of the documentation for self-learners and for the workshops.

We all share a Google Drive folder, where we all share notes about our research and development of the library and the educational material.

2.11 Code

This thesis is distributed as a repository, hosted on the GitHub platform, and available at https://github.com/montoyamoraga/tiny-trainable-instruments.

The auxiliary files, such as the LaTeX project for this document, and the auxiliary Jupyter notebooks, and documentation and tutorials are included on this repository.

The main software component of this project is the TinyTrainable library, available at https://github.com/montoyamoraga/TinyTrainable and also through the Arduino IDE.

The code included on this library is distributed on the folders:

1. examples/
2. src/

2.11.1 src/

The source code for where there is a TinyTrainable.h and TinyTrainable.cpp file where we included all the basic functionality of the library. Additional subfolders include

inputs/

Base class Input and inherited classes for each one of the other inputs.

outputs/

Base class Output and inherited classes for each one of the other outputs.

tensorflow/

Auxiliary files, copied from the examples from the Arduino TensorFlow Lite that we are building on top of, and also from the newer TinyML library by the EdX team. These, unless otherwise noted, are included without modifications and distributed through the Apache License included on each file's headers.

2.12 Tools

This is a summary of tools used for making this project.

2.12.1 clang-format

Tool for automation of formatting to source code. More information at `https://clang.llvm.org/docs/.ClangFormat.html`.

2.13 Doxygen

Tool for generating documentation from the source code. More information at `https://www.doxygen.nl/`.

2.13.1 GitHub Actions

Every time we push code to the TinyTrainable repositories, a GitHub action creates a virtual machine, and runs a script to generate the Doxygen documentation and push it to the gh-pages branch, hosted at `https://montoyamoraga.github.io/TinyTrai` `nable`.

2.13.2 Jupyter

Jupyter is a free, open-source browser application that allows users to easily read and write code in a clean, accessible environment. Code is segmented into cells, which users can execute individually by clicking into and selecting the triangle "play" button at the top. Subsequent code executes based on operations done in previous cells. Basically, Jupyter notebooks allow programmers to create clean, step-by-step interactive walkthroughs through their code. More information at `https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/index.html`.

2.13.3 Markdown

Markdown is a lightweight markup language with simple, intuitive syntax. Aside from a few key differences, it is largely the same as plaintext. The documentation of this project is written using Markdown, including this document! More info at `https://guides.github.com/features/mastering-markdown/`

Chapter 3

Early experiments

And you may ask yourself
"Well... how did I get here?"

Once in a Lifetime
Talking Heads, 1981

In this chapter I showcase my journey, experience, pitfalls, breakthroughs, and what led me to working on this thesis, and I will try my best to be open, detailed, celebratory, and critical.

3.1 Learning microcontrollers

I learned how to program microcontrollers around 2010 as an undergraduate student of electrical engineering in Chile using PIC microcontrollers and Microsoft's tools, including the operating system Windows and the C# programming languages. In parallel, with some classmates we discovered the Arduino Uno microcontroller, and with my friend Braulio we built a robotic guitar tuner, where the Arduino detected the pitch of a string, and made a motor move the tuning gear of the string to match the desired pitch.

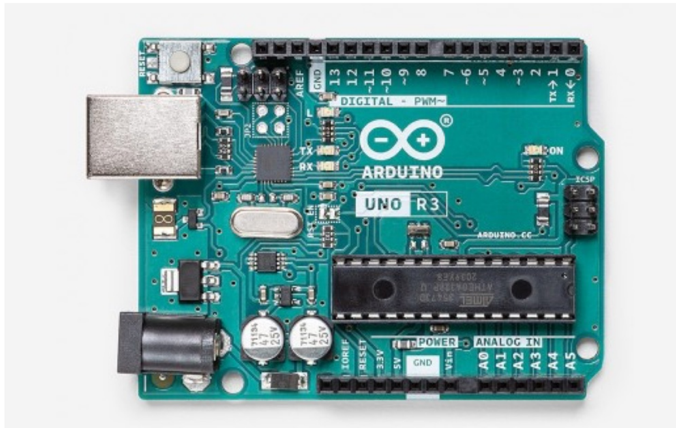


Figure 3-1: Arduino Uno microcontroller

Retrieved from [13]

Fast forward to 2013, for my thesis I had to complete a capstone project and implement many low-level programming techniques, and Arduinos were not allowed because they were considered a shortcut. With my friend Guillermo we built a robotic device with a PIC microcontroller programmed with C#. Our code was very specific to that particular chip and project, and hardly reusable or interesting for a wider audience.

I wasn't excited about the prospect of making one-off devices, with non-reusable code that I couldn't share, so I haven't programmed PIC microcontrollers since then. In contrast, Arduinos became a huge part of my practice since then, because of their low cost, ease of programming, and of the available resources and documentation, and user contributed libraries, including the TinyTrainable library developed for this thesis project.

3.2 Computer music and physical computing

During undergrad I took classes and did research with professors and computer musicians Rodrigo Cádiz and Patricio de la Cuadra. With them I learned the fundamentals of computer music, including languages such as Max and Pure Data, which I still use to this day. For a class project I created a spoon synthesizer with masking tape, cardboard, and a Makey Makey, a device created in 2021 by Eric Rosenbaum and Jay Silver from MIT Media Lab's Lifelong Kindergarten research group. This was my introduction to physical computing, as a way of creating custom playful interfaces for manipulating media with computers for arts.

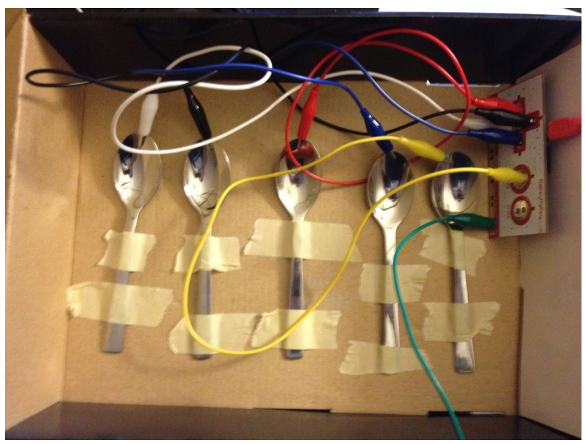


Figure 3-2: Spoons and Makey Makey synthesizer

Picture taken by myself

This was my first time interfacing with my computer without a mouse or a keyboard, but with a flexible instrument that was the result of me tinkering around with cheap reusable materials, in a similar remixable way that I arrange my pedalboards for playing guitar in fleeting ways that are constantly changing.

3.3 Learning open source hardware

After graduation in 2013 I freelanced as a software and technology designer and developer for artists. I learned computer protocols and networks, and wrote custom software for live multimedia theater and music shows. I discovered Processing, a project that Arduino was based on, and I dived deep on computer graphics, interactivity, and it quickly became central to my practice and work.

Realizing that I wanted a bigger community of people to learn media arts with, I researched communities where I thought I could concentrate on learning this craft in a focused and immersive way, so I applied to New York University's Interactive Telecommunications Program (NYU ITP), where I joined as a graduate student in 2015.

In my first semester I took the class Introduction to Physical Computing, taught by one of Arduino's co-creators Tom Igoe. Since I was already familiar with electronics and circuits, I focused on learning interface design, human computer interaction, open source hardware and software, and physical computing education.

During my research I was introduced to a wider ecosystem of microcontrollers beyond Arduino, that were possible because of its open source nature and the community behind it. My favorite example is the Teensy by PJRC, which captivated me by its USB MIDI capabilities, which allowed for standalone MIDI operation, and the creation of plug and play devices that needed no additional setup or scripting. Also by its audio library, which allowed me to create interactive standalone experiences, playing audio samples and applying audio effects.

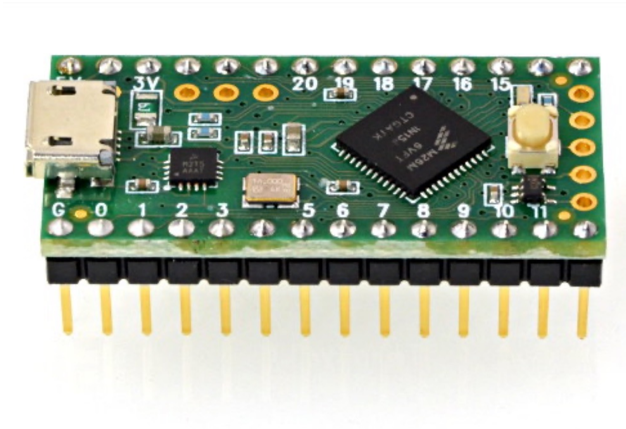


Figure 3-3: PJRC Teensy LC microcontroller with pins

Retrieved from [25]

3.4 Processing and p5.js

The program at NYU ITP is by definition in constant change, and I feel lucky because in my first semester's Introduction to Computational Media class we were taught p5.js instead of Processing, a change that hadn't happened in a year. I had almost no experience in web programming, and I mostly didn't see the point of it.

Lauren McCarthy, the creator of the p5.js library, was a professor at NYU in my first year, and I took her class Performing User, about performance art and technology, and it was my favorite class in graduate school.

At NYU ITP I but mostly about web and scripting, and my thesis concentrated on open source, performance art, with only a small hardware component in the form of a Raspberry Pi computer with a countdown timer to my projected death time, according to data by the United Nations, based on my assigned-at-birth-gender and my birth place.

3.5 Teaching

After graduating from NYU ITP I focused on media arts education, writing tutorials, teaching introduction to programming workshops for artists.

When I joined MIT Media Lab in 2019, I made the conscious decision of focusing on hardware, to give my creations a life outside of my computer, also inspired by newer restrictive developments by Apple, such as restricting the use of apps created by unregistered developers. In my first semester, which ended up being the only on-campus semester I had, I was introduced by my friend Will Freudenheim to the Shbobo synthesizers by Peter Blasser.

3.6 Publishing on the web

3.7 Publishing libraries

UNTIL HERE

I was partially aware of the instruments made by Peter Blasser, in particular the analog ones.

While at MIT Media Lab, I was delighted by the newer versions of Teensy, which are even faster and more powerful, and which led me to start designing handheld samplers for field recordings, and other standalone devices.

This in turn led me to review the current NYU ITP materials for physical computing, where they currently stopped using the now classic Arduino Uno, and have incorporated in their teaching the new series of Arduino Nano microcontrollers, which I based my thesis on.

In particular, the Arduino Nano 33 BLE Sense I am using, comes with 9 sensors, to measure and detect acceleration, movement, distance, color, and a microphone. This is an amazing breakthrough, since now we can use all this data without having to purchase, install, or calibrate the sensors.

I am using the Arduino's sensors to gather multimedia input data, analyze it with machine learning, and then respond with multimedia outputs.

3.8 Artistic machine learning

My first experiment in this field was with my NYU ITP classmate Corbin Ordel, who was a student at Gene Kogan's Machine Learning for Artist class. Together we audited Rebecca Fiebrink's Machine Learning for Musicians and Artists class, available at the Kadenze platform, and learned the Wekinator platform.

Our project is Piano Die Hard, a digital sculpture consisting on a piano that reacted in real time,

teamed up to hack a project we called Piano Die Hard, built with open source tools such as Wekinator, Arduino, openFrameworks, and using the machine learning algorithm KNN. We created a video database of explosions in the Die Hard movie franchise, and another one of other 1980's movies with no explosions, and we trained our machine learning algorithm to distinguish between the categories explosion and no explosion. We featured this project at a NYU ITP show, were written up at the Daily Beast newspaper, and exhibited our work at the alt-ai conference.

In 2017, while I was finishing my appointment as research resident at NYU ITP, Cristóbal Valenzuela had started the project RunwayML as his master's thesis, which is now a company led by Cristóbal, Alejandro Matamala and Anastasis Germanidis.

At NYU ITP I also saw the first experiments with deeplearn.js, later TensorFlow.js, which soon became the foundation of the ml5.js library, a wrapper for TensorFlow.js, for on-the-browser machine learning.

I decided I wanted to dip my toes in machine learning, so I took a month-long intensive class at the School of Machines in Berlin, Germany, facilitated by Gene Kogan and Andreas Refsgaard, and organized by Rachel Uwa.

A big inspiration for this thesis has been the book on GANs by Casey Reas, published by Anteism, as of 2021 on its second edition. It's an arts-first book that contextualizes the use of machine learning algorithms for the creation of images, and uses the metaphor of these algorithms as being similar to the development of the camera. Artists don't need to understand all the physics or mechanics behind a camera in order to make art with it, but it can help to understand it too. I think machine learning is also a game changer for instrument making, and machine learning introduces new civic complexities, and my thesis tries to follow the example of this book, to introduce the technology and contextualize for a new generation of artists and instrument makers.

Since many machine learning projects rely on proprietary hardware, such as NVIDIA GPUs, or rely on the cloud for faster compilation times, for this thesis I decided to make open source machine learning projects that people could read and understand and remix and hack.

TODO: mention the impact of the documentary Coded Bias, and how these researchers impacted my desire to make my thesis. Also mention how right before pandemic I had started a pottery class, with the intention of making clay-based instruments for thesis, as a metaphor of making code and hardware and software feel fluid and not static, I want to empower people to program, in particular machine learning because of its dangerous implementations by oppressive governments and corporations, and in particular for arts, for making artists dream come true.

Chapter 4

Background and inspiration

It has to start somewhere
it has to start sometime
what better place than here?
what better time than now?

Guerrilla Radio
Rage Against the Machine, 1999

4.1 Research at MIT

As part of the research that directly inspired this thesis, here are some courses I took and some projects I started while at MIT Media Lab, and how they inform my theoretical and practical background for Tiny trainable instruments.

4.1.1 Classes at MIT

1. Fall 2019, CMS.901 Current Debates in Media, by professor Sasha Costanza-Chock
2. Fall 2019, MAS.S65 Recreating The Past, by professor Zach Lieberman
3. Spring 2020, MAS.826 Sound: Past and Future, by Tod Machover
4. Spring 2020, MAS.712 Learning Creative Learning, by professor Mitchel Resnick

In the class Current Debates in Media, topics covered included fake news, surveillance, algorithmic bias, data colonialism, climate justice, algorithms of oppression, among others. For my final paper I wrote on the role of the media during the 2019 Chilean protests. This class directly inspired me to look for privacy-first machine learning, and thinking about artificial intelligence in a critical way and not as clean safe automation, but as invisibilized labor, exploitation, and algorithmic bias and discrimination in the worst scenarios.

In the class Recreating the Past, I learned about media arts history, and I dived deep in the language C++ which I used for writing the TinyTrainable library, while working with the library openFrameworks, which is one of the most popular open source frameworks and communities for media arts.

In the class Sound: Past and Future, I learned more about the history of different computational advancements for sound, with a strong focus on projects at MIT Media Lab's own research groups including Opera of the Future, Hyperinstruments, and Music, Mind, and Machine. This class introduced me to many projects and it helped me decide on making instruments for my thesis, using the latest technologies I could find, in this case, microcontrollers and machine learning.

In the class Learning Creative Learning, I was introduced to the Lifelong Kindergarten's frameworks and ideas, including the 4 P's (peers, projects, passion, and play), and the design of Scratch as a home with low floor, wide walls, and high ceiling, which I highly recommend to check on Mitchel Resnick's book Lifelong Kindergarten. This class gave me the push to write a library with a community in mind, starting with the development of it, which happened because I had the pleasure of working with MIT undergrads Peter Tone and Maxwell Wang, who helped me with different aspects of research and development.

4.1.2 Projects at MIT

Some other projects I created during this master's include:

1. SiguesAhi: an instrument to detect when oppressive institutions have ceased to exist. It is achieved with microcontrollers with Internet connectivity.
2. Open Drawing Machine, with Gaurav Patekar: an open source low cost programmable drawing machine
3. Introduction to networks for artists: a series of tutorials for beginners, to learn how to set up their own networks and collaborate in peer-to-peer ways for making art.

SiguesAhi is a sibling project to Tiny trainable instruments. It is based on a microcontroller from the same series and format but different architecture and capabilities, the Arduino Nano 33 IoT.

For this project, I am writing a library for making digital instruments that can detect if institutions still exist. This is accomplished by connecting the microcontroller to the internet, and making it ping the Wikipedia website with a certain periodicity, in order to download and check the first sentence of the first paragraph of the entry. As an example of the library, I am using the National Rifle Association, which sadly still exists.

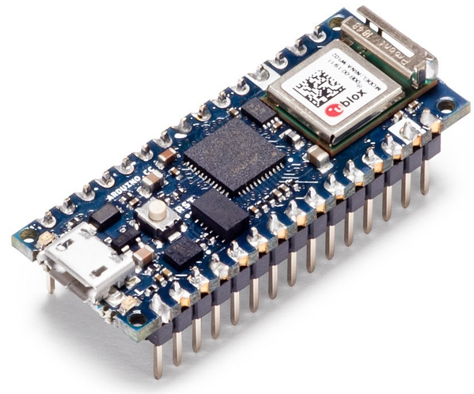


Figure 4-1: Arduino Nano 33 IoT with headers

Retrieved from [12]

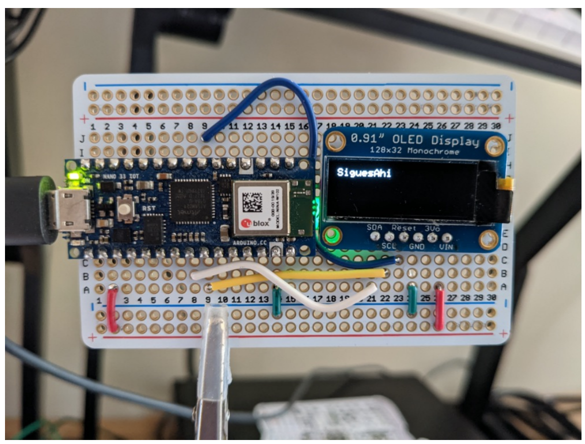


Figure 4-2: SiguesAhi project

Picture by myself

The [14, Open Drawing Machine] is a collaborative project with Gaurav Patekar for the research group Future Sketches at MIT Media Lab. It is a low cost open source machine, consisting of an Arduino microcontroller and hardware for drawing computationally. We have done the research together, Gaurav designed and 3D printed the hardware, and I have packaged the results as a still unfinished openFrameworks addon.

Currently the Open Drawing Machine can receive drawing commands from a computer through a serial port, and the next step of this project is being able to receive

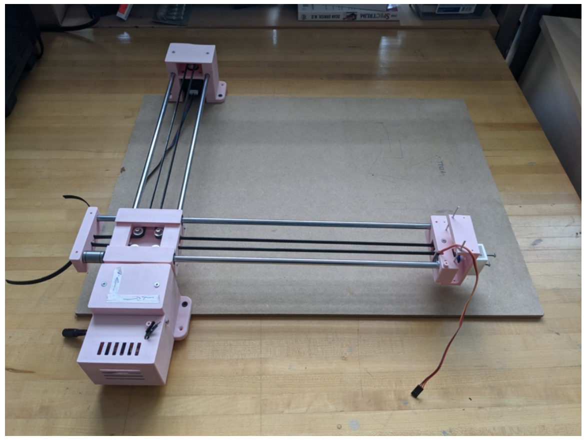


Figure 4-3: Open Drawing Machine project

Picture by myself

commands from other computers through networks and the internet. This comes from the last project I will mention that I worked at while at MIT Media Lab, which is Introduction to computer networks for artists [20], a collection of tutorials for remote peer-to-peer collaboration between artists. This project is inspired by the amazing research on remote collaboration, sensors, peer-to-peer protocols, by friend Lisa Jamhoury, including the app Kinectron

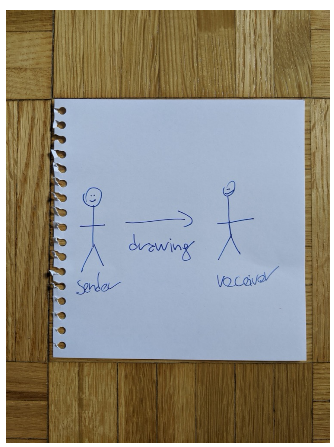


Figure 4-4: Introduction to computer networks for artists project

Picture by myself

4.2 Computational media arts instruments

On top of the proliferation of personal portable computers capable of performing real-time audio, and their creative live use by compute

In particular, here I will highlight some media arts instruments that have inspired my research, because of their use and promotion of open source software and hardware, scripting capabilities, and other design considerations. These instruments often sit at my desk for inspiration, or I spend hours playing with them for my art and learning from them and the communities around them.

The tables 4.1 and 4.2 are respectively a technical and influences summary of the instruments that I reference in this section.

Company	Instrument	Year	Basis	Software
Bastl Instruments	Illuminati	2019	MCU	None
Bastl Instruments	Kastle Drum	2020	MCU	Arduino, C++
Bastl Instruments	Kastle v1.5	2017	MCU	Arduino, C++
Bastl Instruments	OMSynth	2016	IC	None
Bastl Instruments	microGranny 2	2016	MCU	Arduino, C++
Bastl Instruments	Servo	2016	MCU	Arduino, C
Critter & Guitari	Organelle	2016	Linux	Pure Data
Critter & Guitari	EYESY	2020	Linux	Python, Pygame
monome	aleph	2013	Linux	C
monome	norns	2018	Linux	Lua, SuperCollider
Shbobo	Shnth	2013	MCU	Shlisp
Shbobo	Shtar	2017	MCU	Shlisp

Table 4.1: Technical details of media arts instruments

4.2.1 Bastl Instruments

Bastl Instruments is a Czech company of multimedia instruments, which has had a huge impact and influence on my research and practice. When I first started researching the Eurorack format some years ago, I visited the shop Control in Brooklyn NY, and some modules by Bastl stood out to me, because of their wooden panels and

Company	Instrument	Influence
Bastl Instruments	Illuminati	Output with LEDs
Bastl Instruments	Kastle series	Use of breadboard and jumper wires
Bastl Instruments	OMSynth	Distribution as tutorials + parts kit
Bastl Instruments	microGranny 2	Arduino as basis for instrument
Bastl Instruments	Servo	Output with servo motor
Critter & Guitari	Organelle	Computer for sound
Critter & Guitari	EYESY	Output with screen
monome	aleph, norns	Computer for sound
Shbobo	Shnth, Shtar	Input with multiple gestures

Table 4.2: Influence of media arts instruments

interaction with classic physical computing educational materials, such as motors and solenoids, which was an inspiration for me to include support for servo motors in Tiny trainable instruments.

Figure 4-5: Bastl Instruments Servo module

Retrieved from [17]

Another inspiration comes from their microgranny 2 granular sampler which is made with an Atmega microcontroller and its firmware is open source and available as a repository on their GitHub account, along with many other of their instruments.

Their Kastle synthesizers are also based on microcontrollers, and feature a patchbay for making connections with jumper wires, the same used for prototyping in electronic breadboards, which influenced me to make the Tiny trainable instruments built with breadboards and jumper wires, instead of custom printed circuit boards. Also, the

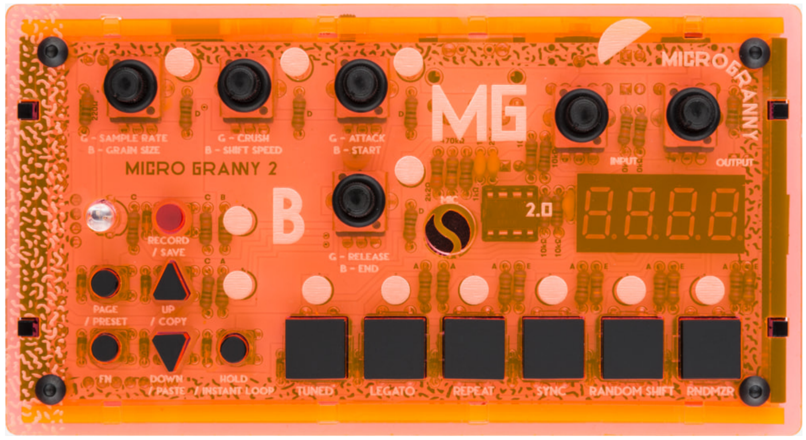


Figure 4-6: Bastl Instruments microGranny 2

Retrieved from [17]

Kastle synths are forgiving instruments, their inputs and outputs are robust enough to allow for mistakes in connections, in an electrical and mechanical way, which I think it's perfect for safe experimentation, it would be a bummer if the instrument was easy to break, or if it demanded a huge effort in understanding electronics for using it.

Figure 4-7: Bastl Instruments Kastle v1.5

Retrieved from [17]

As of writing, 2 different units are in production, both retailing for 100.00 USD, the Kastle v1.5 melodic / drone synthesizer, and the Kastle Drum, for rhythm. The only difference between these synthesizers is the firmware and the labels on the faceplate. The community is encouraged to write new firmware to modify their behavior.

Figure 4-8: Bastl Instruments Kastle Drum

Retrieved from [17]

Another instrument I want to highlight is the Illuminati, currently discontinued, a device that uses different inputs (audio, control voltage, MIDI messages), to control the light intensity of connected USB lamps, which influenced the conception of Tiny trainable instruments as multimedia arts instruments, not only focusing on audio and music, but also printed text, light, and screen output.

Figure 4-9: Bastl Instruments Illuminati

Retrieved from [17]

The final instrument from this company that I want to highlight is the OMSynth, one of many collaborations with Casper Electronics. This device is an educational and maker circuit development tool for creating synthesizers, it includes basic fundamental blocks, such as battery power, audio input and output, potentiometers for attenuating and boosting signals, and a suite of parts kits for building devices including sequencers, oscillators, and samplers, on the included breadboard. Its release as a kit was also a direct influence in the release of Tiny trainable instruments as a kit with instructions.

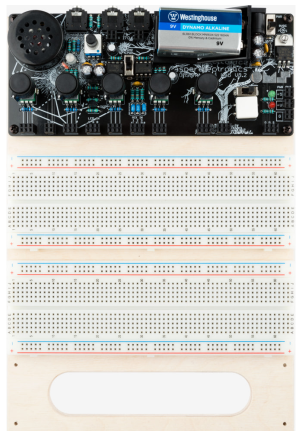


Figure 4-10: Bastl Instruments OMSynth

Retrieved from [17]

Many BASTL standalone instruments are 200.00 USD or less, which is a huge contrast to the 1960s, when a Moog analog system II cost 6,200.00 USD, which was enough to buy a small house [24]. Also, many of their instruments are sold as kits for building and soldering them yourself, for the cheaper cost and the added educational aspect of having hands-on experience.

4.2.2 Critter & Guitari

Critter & Guitari is an American company based in Brooklyn NY, which have released in the past computational microcontroller-based audiovisual instruments, from which my favorite is the Kaleidoloop, currently discontinued. It is a sampler with an internal mic and speaker, that allows you to record audio and then control its output with 2 knobs for volume and playback rate. It is designed to be portable for doing field recordings, and it was an influence on the design of the Tiny trainable instruments library, which allows the construction of standalone instruments, that with a USB power bank or a battery, you can take for a walk or place anywhere you want.

Figure 4-11: Critter & Guitari Kaleidoloop

Retrieved from [15]

This company has released standalone scriptable computers for arts, which run Linux operating system + Pure Data software.

Organelle computer for sound, scriptable, Linux operating system + Pure Data software.

Figure 4-12: Critter & Guitari Organelle M

Retrieved from [16]

ETC and EYESY computers for visuals, scriptable, Linux operating system + Python / pygame environment or openFrameworks.

They can run on power supplies, and are also portable by the use of batteries.

Figure 4-13: Critter & Guitari EYESY

Retrieved from [16]

4.2.3 monome

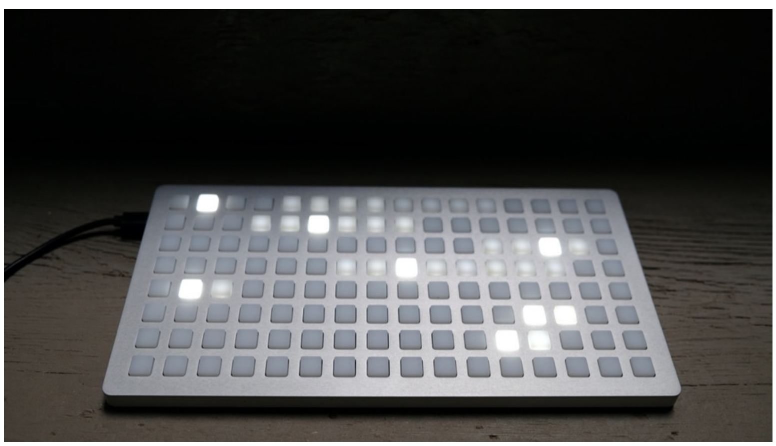


Figure 4-14: monome grid

Retrieved from [19]

Aleph: earlier sound computer.

Norns: sound computer, currently on its second iteration, with expanded hard drive. Also there is a DIY version which is cheaper and runs on a Raspberry Pi. Norns is a Linux machine, running SuperCollider for the sound engine, and Lua scripts.

Norns: sound computer, currently on its second iteration, with expanded hard drive. Also there is a DIY version which is cheaper and runs on a Raspberry Pi.

Figure 4-15: monome aleph

Retrieved from [19]

Norns is a Linux machine, running SuperCollider for the sound engine, and Lua scripts. It has spawned a community that continually releases new scripts and software updates.

Figure 4-16: monome norns

Retrieved from [19]

4.2.4 Shbobo

Peter Blasser has released several collections / companies of musical instruments, the most famous one being Ciat-Lonbarde. Peter also runs Shbobo, which to date has two different instruments, the Shnth and the Shtar.

Figure 4-17: Shbobo Shnth

Retrieved from [26]

Figure 4-18: Shbobo Shtar

Retrieved from [26]

Both run on microcontrollers, and they use a new proposed language called Shlisp, based on Lisp, and also they can be programmed using the Fish IDE.

As of 2021, their firmware and editor became open source, and it is available at github.com/pblasser/shbobo.

COMMENT ON OPEN SOURCE: something being open source doesn't necessarily mean it is accessible to a wider audience. Is one of the goals of your work to create instruments that are accessible to a wider audience?

They promote computer-centric approaches to making sound, such as the use of integers and metaphors of finite state machines, and also allow for different ways of playing and sensing, such as the use of antennas for detecting hand distance, a microphone for detecting speech and whistling, and wooden bars with piezos for detecting pressure.

TODO: write how this inspired the new interactions i am inventing or appropriating for Tiny trainable instruments, such as a drum machine you can talk to, Alexa spinoff.

4.3 Education

This thesis is inspired by the work of the research group Lifelong Kindergarten at MIT Media Lab, led by professor Mitchel Resnick. In the book with the same title, he builds on Seymour Papert's work, and proposes that educational projects should have "Low floor, wide walls, high ceilings", and that learners thrive when they engage in the 4 Ps: "Peers, projects, passion, play".

In terms of peers, I have been lucky to have been supported by the MIT UROP office and MIT Media Lab, and had the opportunity to work with MIT undergrad researchers Peter Tone and Maxwell Wang. Also, this project was taught in collaborative workshops where people could discuss their ideas with their peers.

In terms of projects, this thesis includes the release of a software library, so that people can make the software their own, and spin-off their own projects. It is also open source so that people can learn from my mistakes and also fork to adapt to their needs.

In terms of passion, this thesis is

In terms of play, this thesis project is not about correct answers, or even excellent classification with machine learning, it's all about finding innovative ways to interact with multimedia material, celebrate the small victories and the big glitches, and iterate over and over again.

4.4 Creative machine learning

COMMENT: what is the main argument of the whole piece and how does each independent part connect to that? you should start with a story from previous experiences that is particularly relevant as to why you were inspired to do this work. think "papert and the gears"

While being a graduate student and research resident at NYU ITP I saw how quick things changed in terms of machine learning. I saw how the project deeplearn.js allowed for people to train and deploy machine learning on their browsers, and how this library was acquired by Google and repurposed as TensorFlow.js, a JavaScript version of their machine learning framework TensorFlow.

In turn, at NYU ITP a team of artists and programmers built the library and community of ml5.js, with the 5 being an homage to p5.js. Technically, ml5.js is a wrapper for TensorFlow.js, in the same spirit that p5.js is a wrapper for HTML5 elements such as the canvas.

Another huge contribution to the landscape of machine learning for arts has been the release of the app Runway, which started as Cristóbal Valenzuela's thesis, and is now a company with Alejandro Matamala and Anastasis Germanidis.

After leaving NYU ITP in 2018, I was a student at the month-long workshop "Autonomous Generative Spirit" taught by Gene Kogan and Andreas Refsgaard at the School of Machines, Make and Make Believe in Berlin 2018. We experimented with quick and cheap methods for machine learning, such as the k-nearest neighbors algorithm using the artist and beginner-friendly app Wekinator by Rebecca Fiebrink, and also more computer-intensive algorithms, which sometimes required proprietary hardware such as NVIDIA graphics cards to be trained in a matter of hours, instead of days or weeks using our computers.

There is a tradeoff between speed and cost, and also between monetary cost and time cost.

Figure 4-19: Sam Lavigne, Training Poses, 2018

Retrieved from [18]

The last spark that led me to this thesis was the release of 2 libraries for machine learning on the Arduino platform: The currently beta version Arduino KNN, which allows for on-device training and resembles my earlier studies with Wekinator, in a more portable and private way, no data leaves the microcontroller, and the whole neural network can be wiped with one click of a button.

At a more complex level, I am also working with the TensorFlow Lite Micro, which I learned from Arduino blogs, and which currently is supported by the hardware Arduino Nano 33 BLE Sense.

COMMENT TO THE ABOVE PARAGRAPH: you may not even need to include the specific details here, but just highlight in larger overviews the types of projects you've worked on or the educational fields that influence your work

In late 2020 and early 2021 I completed the just released series of 3 courses of the TinyML Professional Certificate by Harvard at the online platform edx.org

Newer books and references that this thesis was inspired by include the books "You Look Like a Thing and I Love You: How Artificial Intelligence Works and Why It's

Making the World a Weirder Place" by Janelle Shane (2019), and the book "Making Pictures with Generative Adversarial Networks" by Casey Reas (2019).

Also Yining Shi created a new class at NYU ITP in 2020, at the intersection of machine learning and physical computing.

4.5 Digital rights

Machine learning algorithms need data to be trained on. I think it's a human right to not be surveilled, and I hope my thesis can put a positive spin on the gathering of data, by letting users perform auto surveillance, like the Ai Weiwei piece WeiweiCam, a 2021 project where the artist installed cameras for self surveillance as a protest against the Chinese government.

A huge inspiration for my thesis has been the Guardian Project by the Electronic Frontier Foundation, and the research and activism work by Sasha Sasha Costanza-Chock.

4.6 Machine learning

ml5.js

Runway

TinyML Professional Certificate HarvardX

4.7 Digital rights

Electronic Frontier Foundation

Edward Snowden

Design Justice Network

4.8 Opera of the Future projects

During the development of this thesis, I have been fortunate to collaborate on different capacities with other thesis projects by classmates at Opera of the Future, which has directly inspired my work.

4.8.1 Squishies, by Hannah Lienhard

Squishies is Hannah Lienhard's master's thesis, and consists of novel squishable interfaces for musical expression. We shared discussions about low-level sound design, code reusability, sound art education, and digital instruments. We were part of a master's thesis working group, facilitated by Roya Moussapour with two other Media Lab classmates, where we workshopped drafts of our thesis. This practice and feedback has been critical in shaping the language and discourse of this thesis document.

4.8.2 Fluid Music, by Charles Holbrow

Fluid Music is Charles Holbrow's PhD thesis. It is a library for library design, documentation for contributors. The design of the interface, documentation, and scope of the thesis were a direct influence on the documentation and API coding style of this project.

Overall thoughts: I think it would be great if your Chapter 3 clearly led you to the design principles you lay out in Chapter 4. So, why is "open" important -> because of the collaborations or arduino libraries you used. Why is "cheap"/"privacy" -> coded bias and ability to tinker, etc.

Chapter 5

Project evaluation

> Yeah you wanted a hit
> but tell me
> where's the point in it?
>
> ---
> You Wanted a Hit
> LCD Soundsystem, 2010

5.1 Digital release

This thesis project lives on 2 different repositories, hostd on GitHub, to foster collaboration via issues and pull requests, and also with frequent updates, to show people the whole process behind this thesis project.

The main repository contains this thesis document, Jupyter notebooks for machine learning, documentation for beginners and educators, and auxiliary shell scripts. It is hosted at `https://github.com/montoyamoraga/tiny-trainable-instruments`.

The auxiliary repository contains the Arduino library, including its source code and examples. It is hosted at `https://github.com/montoyamoraga/TinyTrainable`.

5.2 Workshop

Figure 5-1: Workshop packages

Picture by myself

For user testing and sharing this thesis, some workshops were conducted during TODO, with support from a grant at CAMIT for teaching the workshops in English in USA, and in Chile in Spanish, remotely over teleconferencing software and after being approved by the MIT COUHES TODO explain.

The workshop instructions are documented on the docs/ folder of the repository available at `https://github.com/montoyamoraga/tiny-trainable-instruments`

Each workshop consists of 2 sessions of 2 hours each, spread over 2 consecutive days.

In the first session we will first help people with installation of the software, and then move on to start wiring the materials on the electronic breadboard material. We will concentrate on the simpler examples with color input. We will also collect data of gesture and speech to create custom databases and use them to train other slow machine learning models that will keep on running on the student's workshops after the workshop is over.

In the second session we will use the result of the trained models to create more advanced instruments that react to gesture and speech. We will also show the participants the other

I applied to and was awarded a grant by the Council for the arts at MIT (CAMIT), which consisted of 2,000.00 USD to buy materials to teach the Tiny trainable instruments workshop to 20 people, during June 2021.

I taught this workshop 3 times, 2 of them were in English for people based in the USA, and 1 time in Spanish for people based in Chile, with a total of 20 kits being delivered.

Figure 5-2: Workshop flyer cover in English

Graphics by Renata Gaui

The multimedia aspect of this project was featured, in particular the ability to use different inputs, including color, gesture and speech, to control different outputs, including serial messages, sound, and motor movement.

Figure 5-3: Workshop flyer cover in Spanish

Graphics by Renata Gaui

5.3 Multimedia documentation

TODO: upload a collection of examples made by people who came to the workshops, featuring the software library and what they learned.

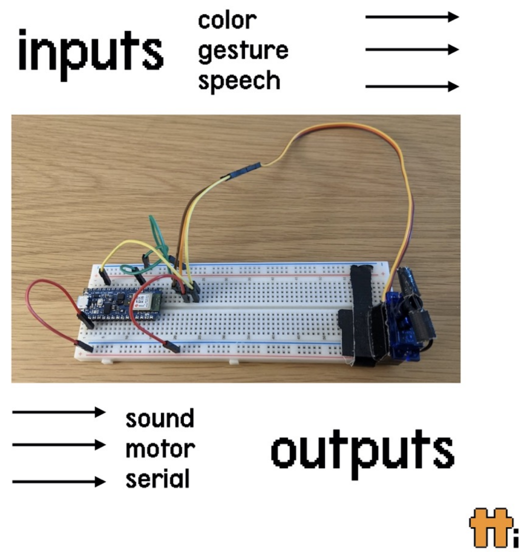


Figure 5-4: Workshop multimedia output in English

By Renata Gaui

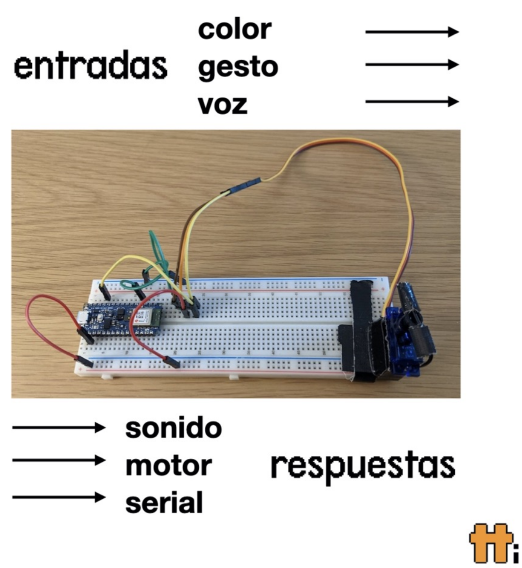


Figure 5-5: Workshop multimedia output in Spanish

By Renata Gaui

Chapter 6

Conclusions and future work

> Don't get any big ideas
> they're not gonna happen
>
> Nude
> Radiohead, 2007

In this thesis I have presented all the stages of the design and development of new standalone multimedia instruments using machine learning and microcontrollers, with a strong focus on AI ethics. The project includes software examples, hardware suggestions, educational material, and strategies for ethical off-cloud machine learning and creation of custom artisanal databases.

This thesis is also the basis for further research, including the creation of subsequent multimedia instruments and software libraries, the writing of new courses and educational units at the intersection of arts, physical computing, interaction design, and computational ethics.

6.1 Contributions

Concrete contributions:

1. Publishing TinyTrainable, a software library for machine learning with microcontrollers for multimedia art.

2. Design, writing, and teaching a 4 hour workshop for beginners, enthusiasts, and artists to teach with the software library.

3. Publishing code and tutorials for creating custom databases for gesture and speech.

4. Publishing custom trained machine learning models for gesture and speech recognition.

5. Publishing code and tutorials for training machine learning algorithms on the cloud and on personal computers for privacy and agency.

6. Publishing other related software libraries, such as MaquinitasParams for communication with other instruments, and MaquinitasRitmos for rhythmic data.

Abstract contributions:

1. Demonstrating how a broader range of people can use machine learning to support their creative expression.

6.2 Lessons learned

1. Writing software for artists is hard.

2. Writing software libraries for other artists is even harder.

3. Collaboration with other people is essential to write code and educational material.

4. Documentation of all design decisions is key to explain why and how everything works.

5. Navigating licenses and copyright is hard.

6.3 Future work

6.3.1 Hardware for new instruments

This thesis relies on an Arduino microcontroller because of their open source nature, commercial availability, software and community support, and detailed documentation.

In particular I picked the Arduino Nano 33 BLE Sense, because of 2 main reasons at the time this project started in late 2020: it is the only Arduino supported by TensorFlow Lite for microcontrollers and featured in the HarvardX certificate on Tiny Machine Learning, and also because of its convenience of having embedded sensors, which makes it simpler and cheaper to acquire data for live interaction and for building custom databases, eliminating barriers to instruments makers and prototypers.

Microcontrollers come and go, most probably this board will be discontinued, but the strategies and software can be forked and adapted to other microcontrollers and software architectures. I am particularly looking forward to adapt this thesis project to other open source microcontrollers, including other Arduino boards, PJRC Teensy and Adafruit Circuit Playground, which would enable the adoption of other software stacks, such as Python instead of C++, and also to other communities building multimedia instruments and experiences.

In terms of the outputs of the Tiny trainable instruments, I focused on creating many parallel multimedia approaches, including making sounds with piezo buzzers and MIDI, manipulating light with LEDs, creating movement and rhythm with servo motors, and printing text with thermal printers and screens. This is to appeal to a larger audience of artists and learners, interested in different mediums, and I hope this thesis project inspires more complex artworks and interactive experiences that this library currently allows, and that people can contribute back to the library to share these new capabilities with everyone.

The Tiny trainable instruments are built with prototyping electronic breadboards, to make explicit their open-endedness, and to promote experimentation and lower barriers. A further iteration of this project would include the creation of custom printed circuit boards with fixed wiring, and also enclosures and packaging.

6.3.2 Software for new instruments

This thesis has been published as an open source software library for Arduino. It promotes modularity and adaptability, where a Tiny Trainable Instrument can be any combination of the multiple inputs and outputs.

The file structure of the source code and the software dependencies of this library was also built with flexibility in mind, to encourage the remix and adaptation of this library to further projects.

A challenging aspect of this project is the breadth of the disciplines combined, and its novel application of machine learning in microcontrollers. As discussed in previous chapters, there is a trend and new wave of builders and makers creating standalone multimedia instruments, based on open operating systems like Linux, and/or different microcontrollers.

Despite the existence of artists and makers building standalone computational instruments, these skills are still hard to acquire. Additionally, the principles of this project, including being as cheap as possible, and as open as possible, are designed to encourage experimentation and hacking, but also can pose additional challenges. I hope this project encourages people to learn how to make instruments, and also engages in discourse about the creation of new curricula for the next generation of instrument makers and artists.

Another challenging aspect of writing software for multimedia instruments is its licensing, both choosing a license and also respecting and understanding the license of other code and resources we are using. The dependencies of this software are mostly other libraries by Arduino, Google, Adafruit, and with different licenses including public domain, MIT, and Apache. I hope this document helps to navigate these legal complexities and that this project helps artists and enthusiasts to navigate this landscape and overcome these barriers.

6.3.3 Educational impact

This project was built to inspire and celebrate a new generation of coursework, workshops, books, in the disciplines of ethical machine learning and microcontroller-based instruments.

I hope that this thesis project is adopted by educators, to introduce students to machine learning, physical computing, media arts, and computational ethics. It would be amazing that aspects from Tiny trainable instruments can be incorporated into existing music, arts, sculpture, and computer science curricula, to create a new wave of instrument makers and media artists.

Appendix A

Context

> You might have to think of
> how you got started
> sitting in your little room
>
> ---
> Little Room
> The White Stripes, 2001

Since joining MIT in summer 2019 I didn't leave the USA, and it's the longest stretch I have had of not visiting my home country Chile. This thesis in particular was written between November 2020 and August 2021, mostly in Boston MA USA, while on a F-1 visa.

A.1 Language

My native language is Spanish, and this thesis was written in English, using the metric system, and the Gregorian year-month-day format.

I tried to avoid violent language, including some widespread conventions on computer science which I hope become obsolete, such as executing a file, instead of running a file.

A.2 Software

This thesis document was written using LaTeX and the Microsoft Visual Studio Code editor, and then exported to the PDF format.

The TinyTrainable library was written in C++, and packaged as an Arduino library, relying on open source library dependencies by Adafruit, Arduino, and Google.

The auxiliary code is a mix of Python scripts, Jupyter Python notebooks, and shell scripts.

The documentation was written using Markdown.

The workshops were taught using the videoconferencing software Zoom, and organized via Google Forms.

A.3 Hardware

This project was written on a 2017 Macbook Air 13-inch, running the macOS 10.15.7 Catalina operating system.

The software library and the software examples were written to be deployed on the Arduino Nano 33 BLE Sense microcontroller.

A.4 Collaborators

Priscilla Capistrano is the senior administrative assistant at the Opera of the Future research group and made sure that everything worked.

Peter Tone is a MIT undergraduate student, who was a researcher, designer, programmer, and tester of the TinyTrainable Arduino library, as part of the MIT UROP program.

Maxwell Wang is a MIT undergraduate student who was a documentation writer, and hardware and software tester, as part of the MIT UROP program.

Roy Macdonald solved my most difficult programming questions, and helped to implement the solutions.

The Council for the Arts at MIT provided the generous funding of the workshop materials.

Renata Gaui designed and created the flyers for the workshops.

Bernardita Moraga packaged and distributed the materials for the workshop in Chile.

Appendix B

Scripts

By pressing down a special key
it plays a little melody

Pocket calculator
Kraftwerk, 1981

For this thesis I developed some scripts which are included in this repository on the scripts/ folder.

I included comments and variables with the hope of making them readable and useful for other people, and also published them on a standalone repository with a MIT License at `https://github.com/montoyamoraga/scripts`.

B.1 Formatting code with clang-format

clang-format is a command line tool for formatting code. This script was written to auto format the code from the Arduino/C++ library TinyTrainable.

```
echo "formatting with clang"
find "$PWD/../TinyTrainable/src" "$PWD/../TinyTrainable/examples" -iname
    "*.cpp" -o -iname "*.h" -o -iname "*.ino" | while read f
do
  clang-format -i "$f"
  echo "formatted $f"
done
```

B.2 Converting formats with ffmpeg

ffmpeg is a command line tool for converting audiovisual files between formats. This script was written to convert audio files from .mp3 to .ogg format for training a database for speech recognition.

```
# clear command line
"clear"

# directory name
DIR_MEDIA="media"

# extension of original files
EXT_ORIGINAL="mp3"

# extension of desired files
EXT_DESIRED="ogg"

# announce start running script
echo "start running " $PWD/$0

# check if files/ exists
```

```
if [ -d "$DIR_MEDIA" ];

# if files/ exists then
then

  echo "success, $DIR_MEDIA/ exists"

  # check if there are .mp3 files in files/
  if [ -f $DIR_MEDIA/*.$EXT_ORIGINAL ];

    # if there are files with $EXTENSION in directory
    then

      echo "success, there are matching $EXT_ORIGINAL files"

      # iterate over every matching file in directory
      for i in $DIR_MEDIA/*.$EXT_ORIGINAL;

        # pipe the filename into cut
        # -d is delimiter of '.'
        # -f is the field number, indexed in 1
        # it retrieves the filename without the extension
        do name=`echo "$i" | cut -d'.' -f1`

        echo convert "$i" $EXT_ORIGINAL to $EXT_DESIRED

        # ffmpeg conversion
        ffmpeg -i "$i" "${name}.$EXT_DESIRED"

        echo converted "$i" to $EXT_DESIRED

        # delete original file
```

```
        rm "$i"

        echo deleted "$i"

      # finish iteration
      done

    # if there are no matching files in directory
    else
      echo "fail, no $EXT_ORIGINAL files in $DIR_MEDIA/"

  # end of if statement for matching files
  fi

# if directory does not exist
else
  echo "fail, $DIR_MEDIA/ does not exist"

# end of if statement for existence of directory
fi

# announce finished running script
echo "finished running " $PWD/$0
```

B.3 Deleting metadata with exiftool

exiftool is a command line tool for reading and writing metadata from files. This script was written to delete metadata from images, like GPS coordinates added by modern smartphones, only keeping the actual image.

```
# clear command line
"clear"

# directory name
# DIR_MEDIA="media"
DIR_MEDIA="$PWD/../thesis/images"

# extension of files
EXT_ORIGINAL="jpg"

# announce start running script
echo "start running " $PWD/$0

echo "looking for files with extension $EXT_ORIGINAL in $DIR_MEDIA/"

# check if directory exists
if [ -d "$DIR_MEDIA/" ];

# if directory exists then
then

  # announce directory exists
  echo "success, $DIR_MEDIA/ exists"

  find "$DIR_MEDIA" -iname "*.$EXT_ORIGINAL" | while read f
    do
      exiftool -all= -overwrite_original "$f"
      echo "formatted $f"
    done

else
  # announce directory does not exist
```

```
  echo "fail, $DIR_MEDIA/ does not exist"

# end of if statement for existence of directory
fi

# announce finished running script
echo "finished running " $PWD/$0
```

B.4 Converting formats with pandoc

pandoc is a command line tool for converting between formats. This script was written to convert from .tex files to .docx files, so that each chapter of this thesis document could be uploaded to Google Docs for feedback from the committee.

```
echo "pandoc latex to docx"

cd "$PWD/../thesis"

# iterate through all .tex files in thesis/
find "$PWD" -iname "*.tex" | while read f
do
  # retrieve basename
  base=$(basename "$f" .tex)
  # delete original docx file
  rm -f "$PWD/docx/$base.docx"
  # create new docx file with pandoc
  pandoc -s -o "$PWD/docx/$base.docx" "$f"
done

# do all the files manually, and with bibliography
```

```
pandoc -o "$PWD/docx/aaron-thesis.docx" "$PWD/chap1.tex" "$PWD/chap2.tex"
    "$PWD/chap3.tex" "$PWD/chap4.tex" "$PWD/chap5.tex" "$PWD/chap6.tex"
    "$PWD/appa.tex" "$PWD/appb.tex" "$PWD/appc.tex" "$PWD/appd.tex"
    "$PWD/appe.tex" --bibliography "$PWD/main.bib"
```

Appendix C

Documentation

Now you play the synthesizer
don't be lazy now
make it hiss like rattlesnakes

'81 How to Play the Synthesizer
The Magnetic Fields, 2017

For this thesis I wrote the following documentation, included at the docs/ folder of the repository, and also included here.

Bill of materials

Notes about the microcontroller

This project is based on the microcontroller `Arduino Nano 33 BLE Sense`. Please don't confuse it with the similarly named `Arduino Nano 33 BLE`! It is recommended to get the one ***with headers***, so you can immediately start using it on a breadboard without needing to solder the headers on it.

Minimum materials

This is the list of minimum required materials for Tiny trainable instruments

Item	Quantity	Cost (USD)	Retailer	Comment
Arduino Nano 33 BLE Sense with headers	1	33.40	Arduino	Microcontroller
Breadboard	1	5.95	Adafruit	Prototyping
Jumper wires	1	3.95	Adafruit	Connections
Micro USB cable	1	2.95	Adafruit	Power

Sound and movement outputs

These are the recommended materials for starters, and are taught in the workshop.

Item	Quantity	Cost (USD)	Retailer	Comment
Piezo buzzer	1	1.50	Adafruit	Output sound
Micro servo	1	5.95	Adafruit	Output movement

Additional outputs

These additional outputs include more expensive or more complex materials, and they are recommended for more advanced users. They are not covered on the beginner workshop, but are supported by the Tiny trainable instruments project and software library.

Item	Quantity	Cost (USD)	Retailer	Comment
LED	1	6.95	Adafruit	Output light
MIDI DIN	1	1.75	Adafruit	Output MIDI data
Thermal printer	1	61.95	Adafruit	Output printed text
128x32 OLED screen	1	12.50	Adafruit	Output screen

Installation

This guide includes information as of June 2021, and we will explicitly include the software versions we are using.

We advise to install the same versions we are using, and if there is any issue with the library or related software, please us know via email or an issue the repository.

For additional documentation, please visit the official Arduino docs website at docs.arduino.cc, and in particular the documentation of the Arduino Nano 33 BLE Sense microcontroller at docs.arduino.cc/hardware/nano-33-ble-sense.

Arduino IDE

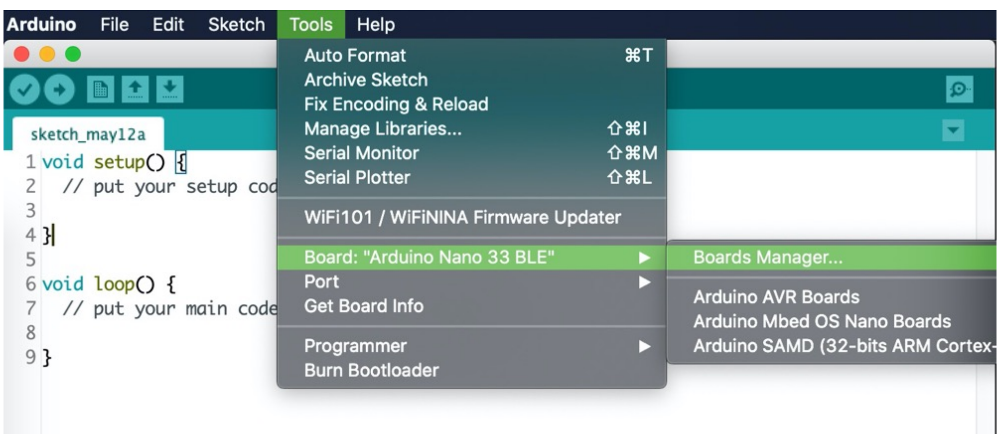
Download and install the Arduino IDE, available at https://www.arduino.cc/en/software. Select the stable release corresponding to your computer's operating system.

As of June 2021, we are using Arduino IDE 1.8.15.

Arduino Mbed OS Nano boards

After installing the Arduino IDE, we need to install the core and necessary libraries for the Arduino Nano 33 BLE Sense microcontroller. Open the Arduino IDE and navigate on the menu to the `Boards Manager`:

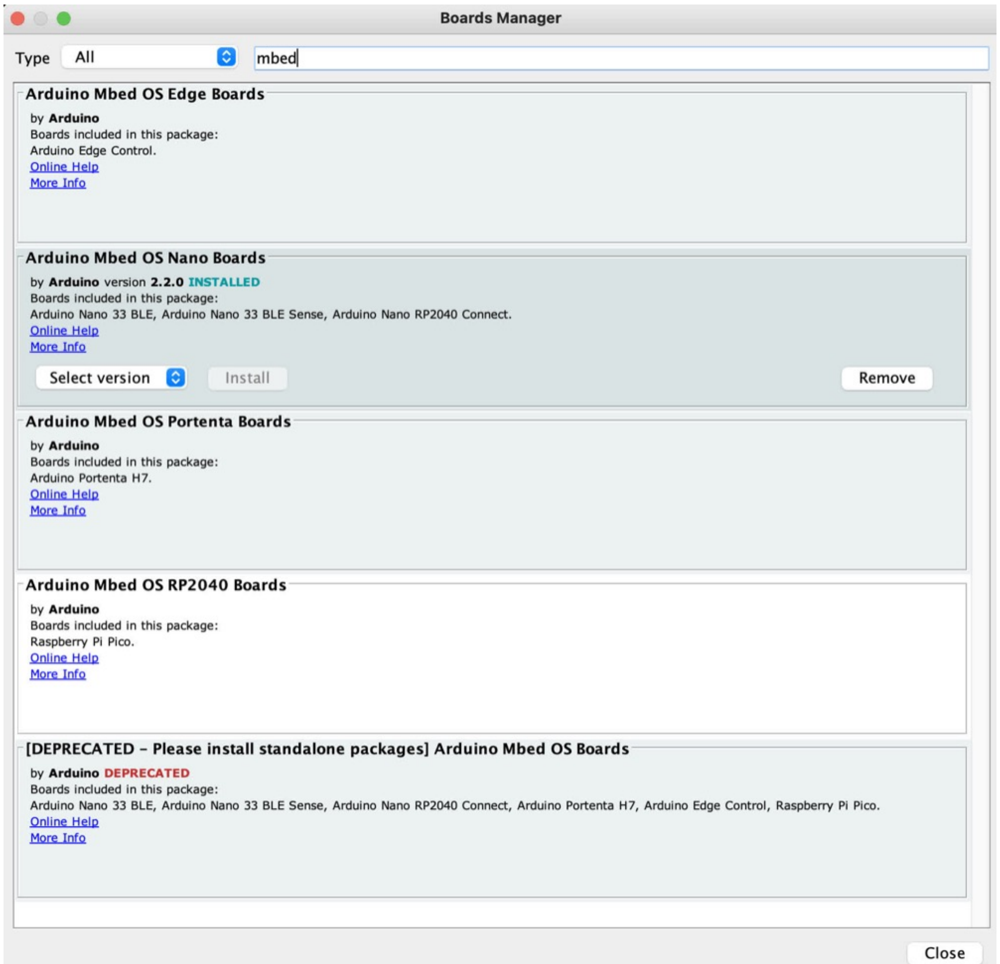
`Tools > Board: "<board_name>" > Boards Manager...`



Use the search bar to find the option `Arduino Mbed OS Nano Boards` and install it, this might take a while.

Please note that if you look for "Mbed", several different options will appear, be careful with the similar named one called `Arduino Mbed OS Boards` which is deprecated and we should not install.

As of June 2021, we are using version 2.2.0.



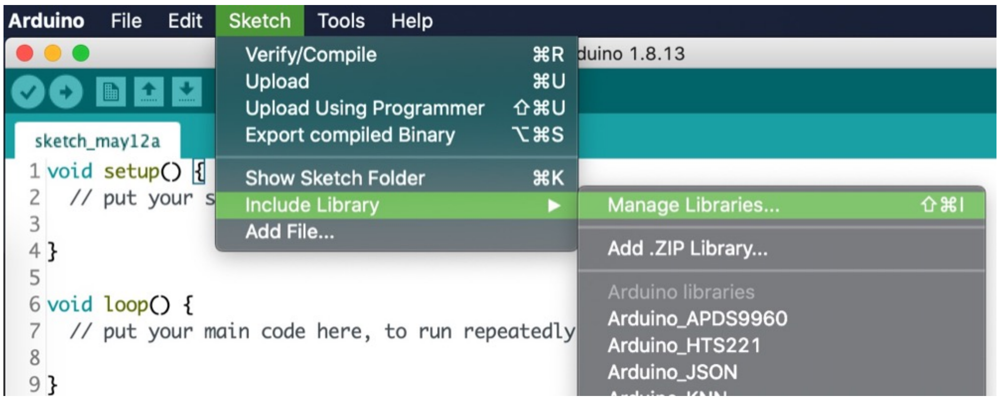
After the installation is complete, we can select the board we are going to work with (Arduino Nano 33 BLE), from the `Tools` menu:

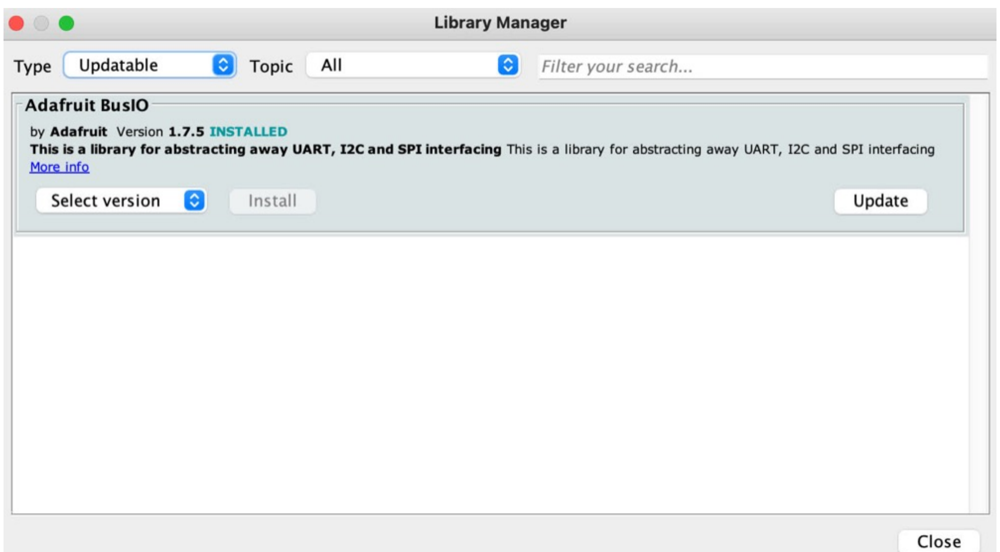
Tools > Board: "<board_name>" > Arduino Mbed OS Nano Boards > Arduino Nano 33 BLE

Please note that this option is valid for both Arduino Nano 33 BLE, and for the board we are using, the Arduino Nano 33 BLE Sense.

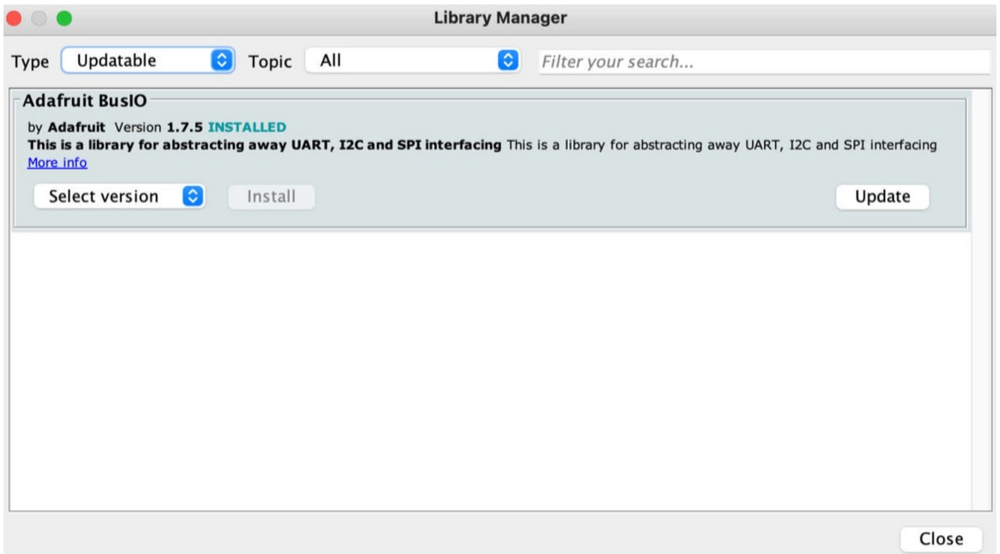
Arduino libraries

Before installing the TinyTrainable library for this project, please first update all your installed libraries. On the Arduino IDE, navigate on the menu to `Tools > Manage Libraries... >`, and then on the `Type` dropdown menu select `Updatable`.



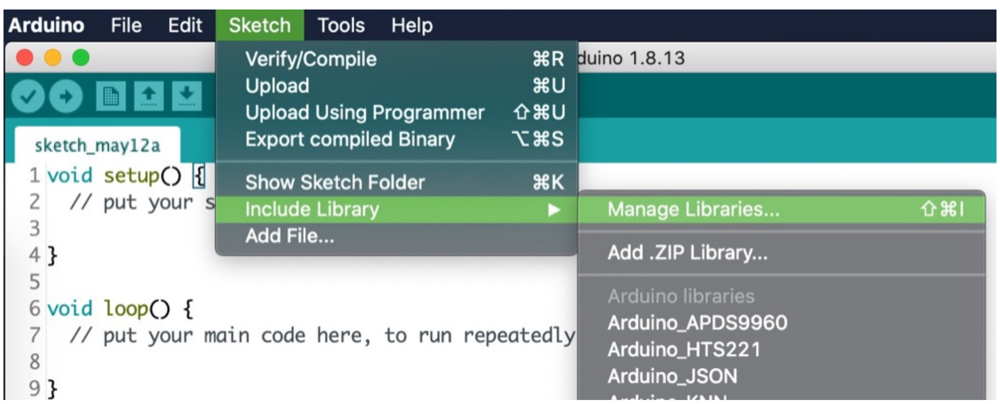


To update each outdated library to their latest version, hover on top of each library, and click on the button `Update`. For this example we are showing the updating of the library Adafruit BusIO, which is installed on my computer, but most probably is not on yours, and you don't need it for this project either.

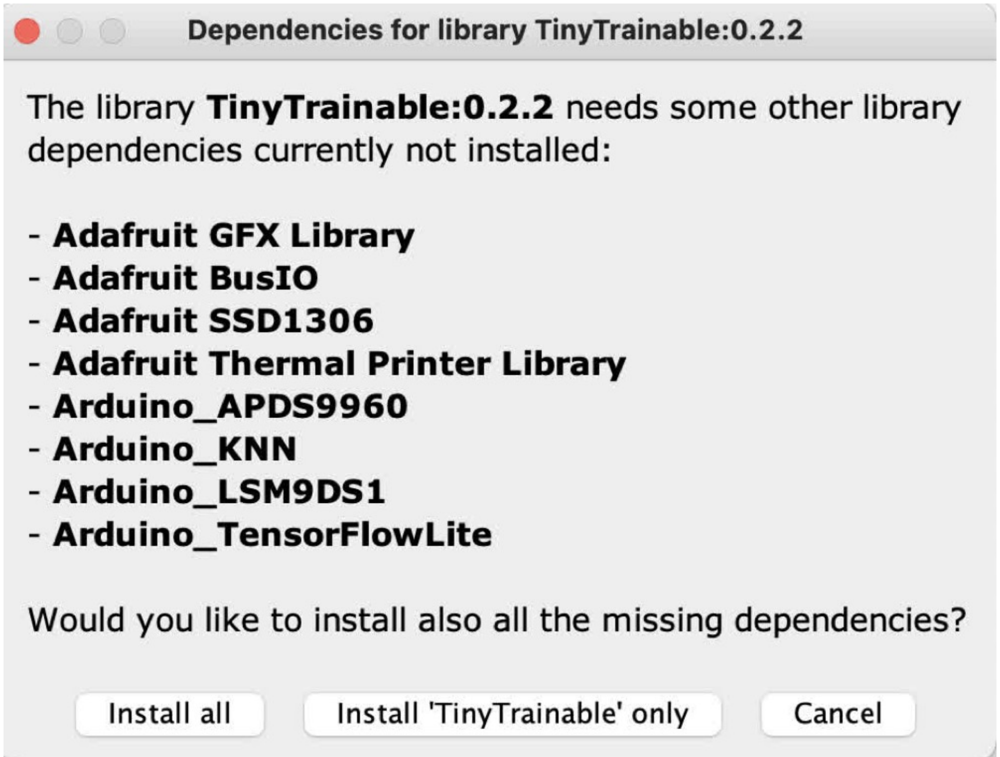


Please repeat this process until there are no updatable libraries left.

Next we will install all the libraries needed for this project. On the Arduino IDE, navigate on the menu to `Tools > Manage Libraries... >`



Go to the search bar of the Libraries Manager and type `TinyTrainable`. This installation will give you the option to also install its dependencies, select `Install all` to download them.



As of June 2021, the latest version 0.2.2 of the TinyTrainable library has these dependencies:

Libraries for using the embedded sensors of our microcontroller:

- `Arduino_APDS9960`: color, proximity
- `Arduino_LSM9DS1` acceleration, magnetic field, gyroscope orientation

Libraries for machine learning:

- `Arduino_KNN`: k-nearest neighbor algorithm.
- `Arduino_TensorFlowLite`: microcontroller version of the TensorFlow machine learning library. Please download the latest non-precompiled version.

Libraries for multimedia output:

- `Adafruit GFX Library`: for output with screen.
- `Adafruit SSD1306`: for output with screen.
- `Adafruit Thermal Printer Library`: for output with thermal printer.
- `Servo`: for output with servo motors.

Python for machine learning

For input-color, you only need Arduino libraries.

For input-gesture and speech, you either need to install specific Python libraries on your computer, or use the free Google Colab service, because we will create databases and train algorithms on a computer.

For beginners, we suggest using Google Colab, because it will be an easier installation, and the algorithms will run faster.

If you decide to run the algorithms on your machine, you will need Python, TensorFlow and Jupyter.

These are the versions we will be using, as of June 2021:

- Python 3.8.6
- TensorFlow 2.3.2
- Jupyter Lab 3.0.5

Your computer might have Python already installed, but it might be one that is not compatible with the TensorFlow version we are using, so we suggest using a Python version manager, like the tool pyenv https://github.com/pyenv/pyenv.

After installing pyenv, open the terminal and go to this repository. If you dont know how to download a repository to your machine, follow this tutorial about cloning repositories from GitHub.

```
cd tiny-trainable-instruments/
```

Check that pyenv is able to read the .python-version file

```
pyenv versions
```

You should see a list, with the version we are using and an asterisk, to highlight that this is the Python version we will use. If there is no asterisk and it says that the required version of Python is not installed, use the command:

```
pyenv install <python version number>
```

If you are using an old version of pyenv, there's a chance that the install won't work; copy the entire command pyenv gives you (including the &&'s) and enter it into the terminal. Then once pyenv is updated, try the above command again.

Now that you have the correct version of Python, create a virtual environment (which we will name env) using the Python package venv. Most dependency problems can be solved by using a virtualenv; we can’t support issues not using a virtualenv due to the huge variety of system configurations. On your terminal type:

```
python -m venv env
```

Activate the virtual environment with this command, which you will use every time you want to enter the venv:

```
source env/bin/activate
```

Now your terminal should have every new line starting with (env). Your command prompt should look something like this:

../docs/images/1-arduino-boards-manager

The pip of your Python virtual environment might need updating; you can update to the latest version with the command

```
pip install --upgrade pip
```

Then use pip to install the Jupyter packages, along with their dependencies:

```
pip install -r requirements.txt
```

Now you can run the Jupyter Lab tool with `jupyter-lab`. This will open a tab on your browser to navigate through the files in your computer and allow you run code and read the documentation.

The code for input-gesture and input-speech is written using Jupyter notebooks, which have the extension .ipynb, and are located on the folder `instruments/`. The documentation is written in several Markdown files with extension .md. These files are on the folder `docs/`, which includes an index on README.md.

If you double click on a Markdown file, it will open an Editor window with the Markdown code. To view the rendered text you can right click and select "Open with Markdown Preview". If you have internet connection, it might be more convenient to access the online documentation on the online repository.

To close the Jupyter notebook server, press `ctrl+c` in the terminal (even on OSX; it's not `cmd`) and confirm with `y`.

To exit the virtual environment once you're done, use the command `deactivate`. Note that the command `jupyter-lab` will not work until you reactive the virtual environment.

Wiring

Conventions

Wires:

- Red = 3.3 V power from Arduino
- Green = Ground from Arduino

Breadboard

Breadboards are built so that within each of the rows, the 5 tie points in the columns labelled `a–e` are electrically connected inside the board and act as a single electrical node, and same with `f–j`.

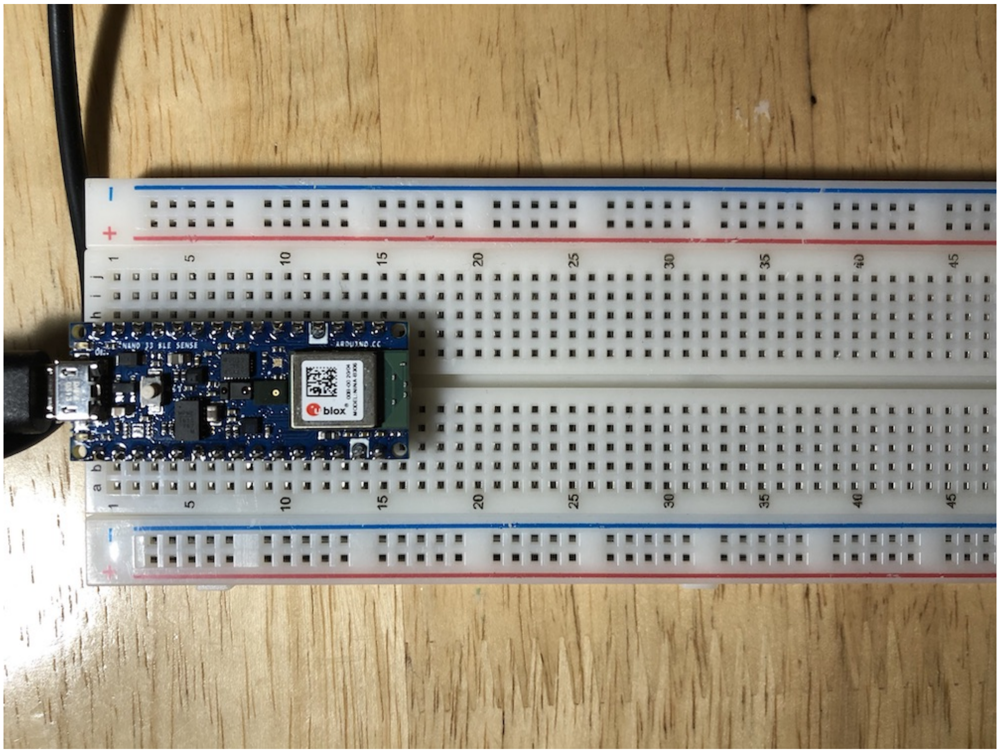
In addition, there are 2 columns to each side of the breadboard, where each column is one electrical node. Conventionally, we connect the positive voltage to the column labelled `+`, and the ground to the column labelled `–`.

A full breadboard guide is available at https://learn.adafruit.com/breadboards-for-beginners/breadboards.

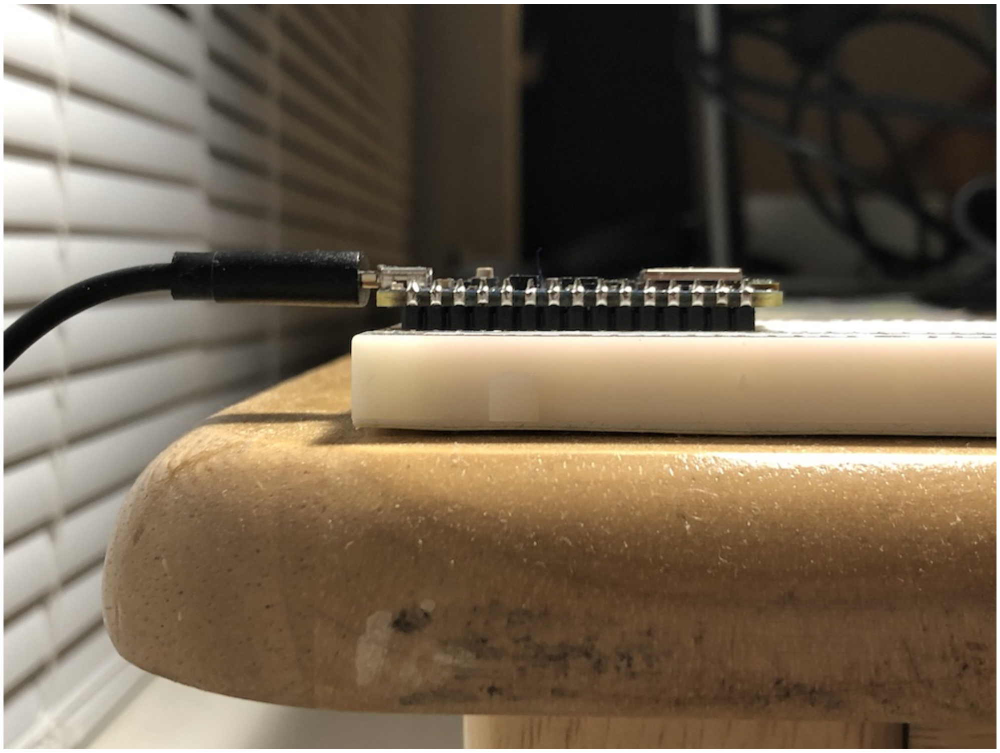
Arduino microcontroller

The Arduino Nano BLE 33 Sense we are using has 30 pins in total, 15 on each side. The official pinout is available at https://content.arduino.cc/assets/Pinout-NANOsense_latest.pdf.

We recommend placing the microcontroller at the top of the breadboard (C1 to C15 and G1 to G15) with the USB Micro port facing up.



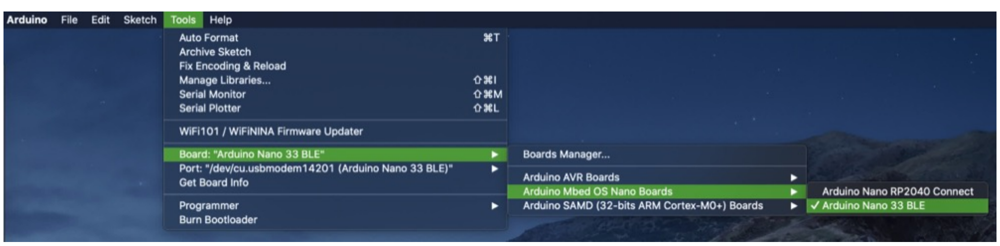
Note that the microcontroller should be flush with the breadboard; none of the headers should be visible.



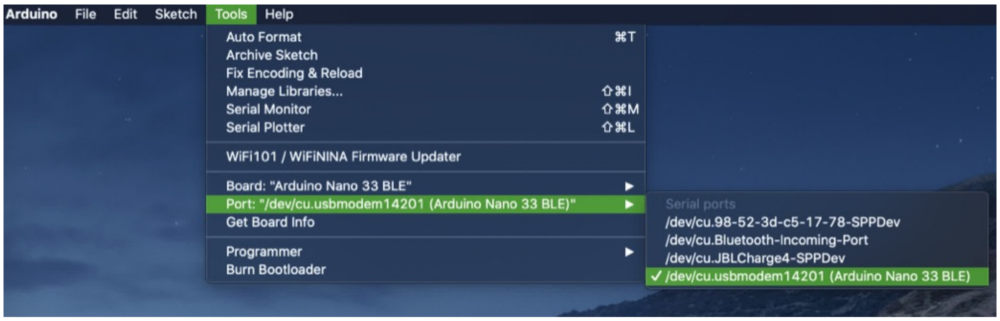
Your first example

Connect your Arduino microcontroller to your computer with the USB cable and open the Arduino IDE software.

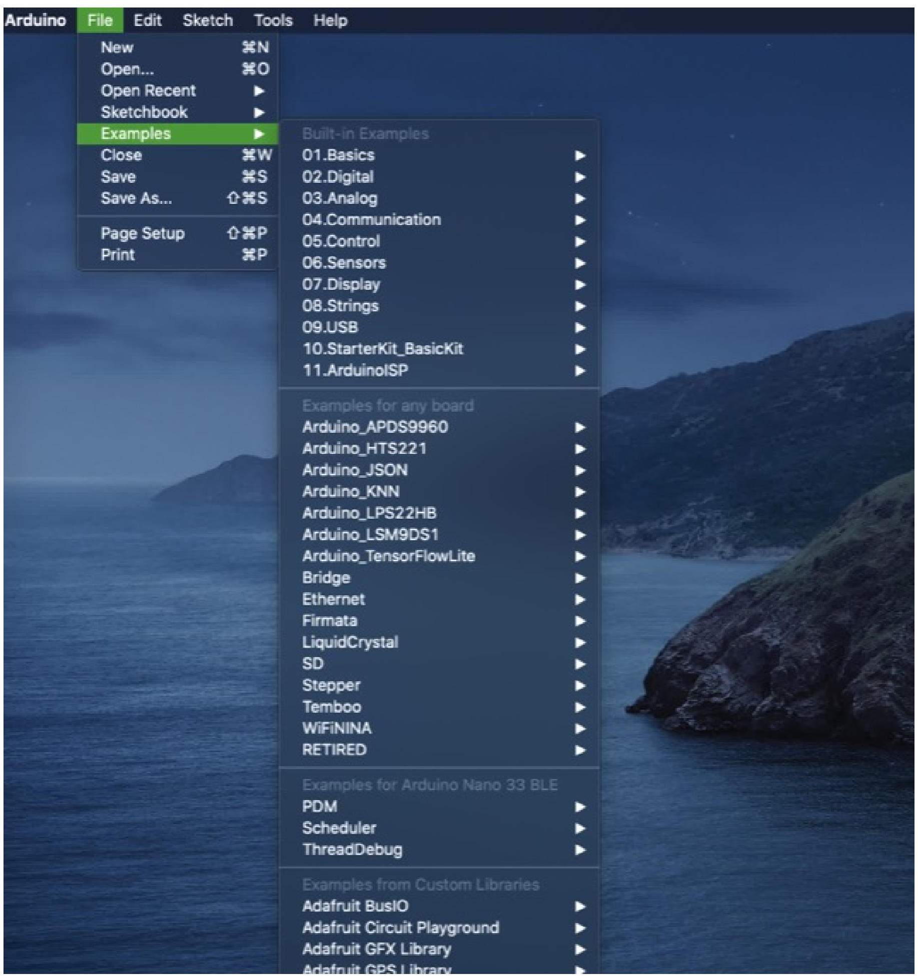
On the board, select the `Arduino Nano 33 BLE`.

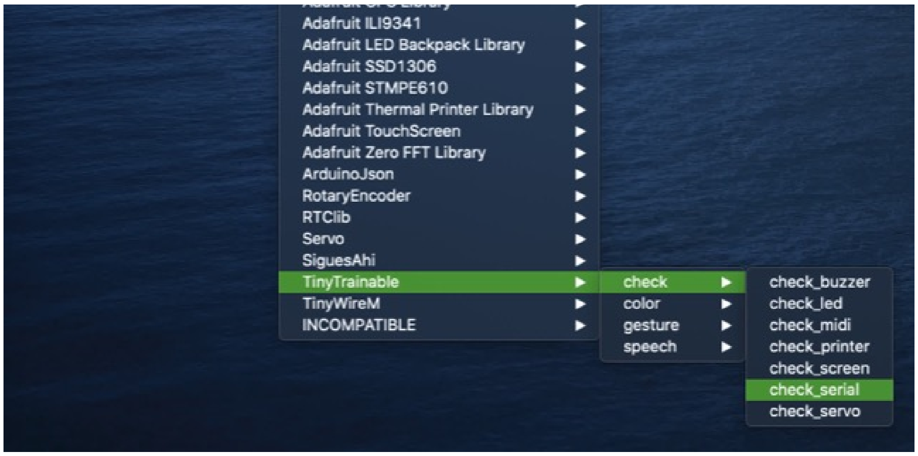


Then make sure yor port points to your Arduino, the number is irrelevant, and the actual text changes between computers.

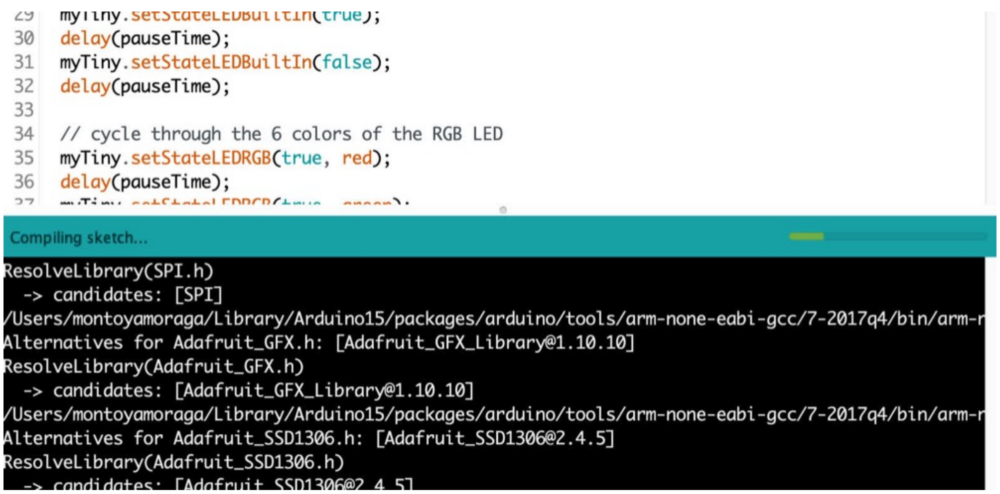


Now let's open the example `check_serial`, included with our TinyTrainable library.

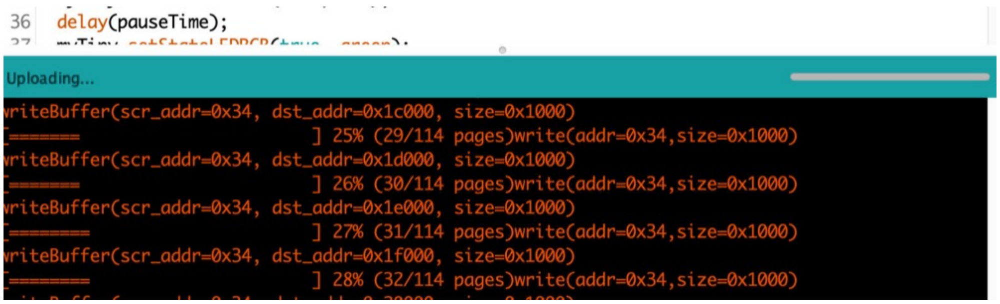




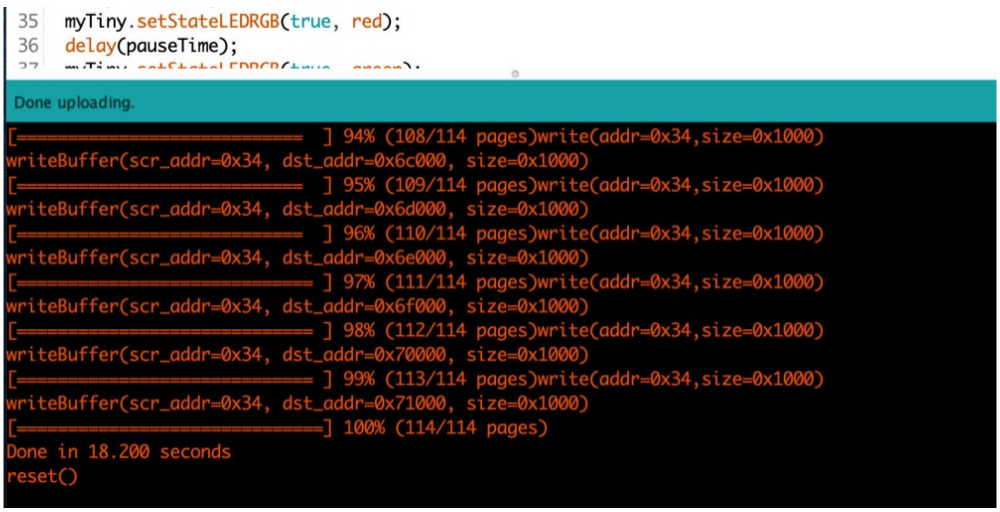
Click on the arrow to the right for uploading the code, which will be shown on the bottom of the Arduino IDE, with the message `Compiling sketch`.



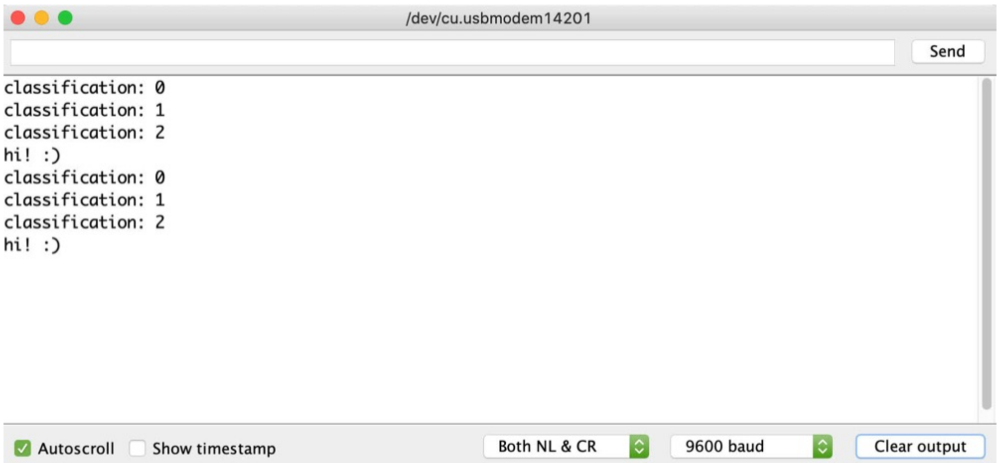
The compilation might take several minutes, and after it is done, the message will change to `Uploading...`



This process is shorter, and after it you will see the message `Done uploading`.



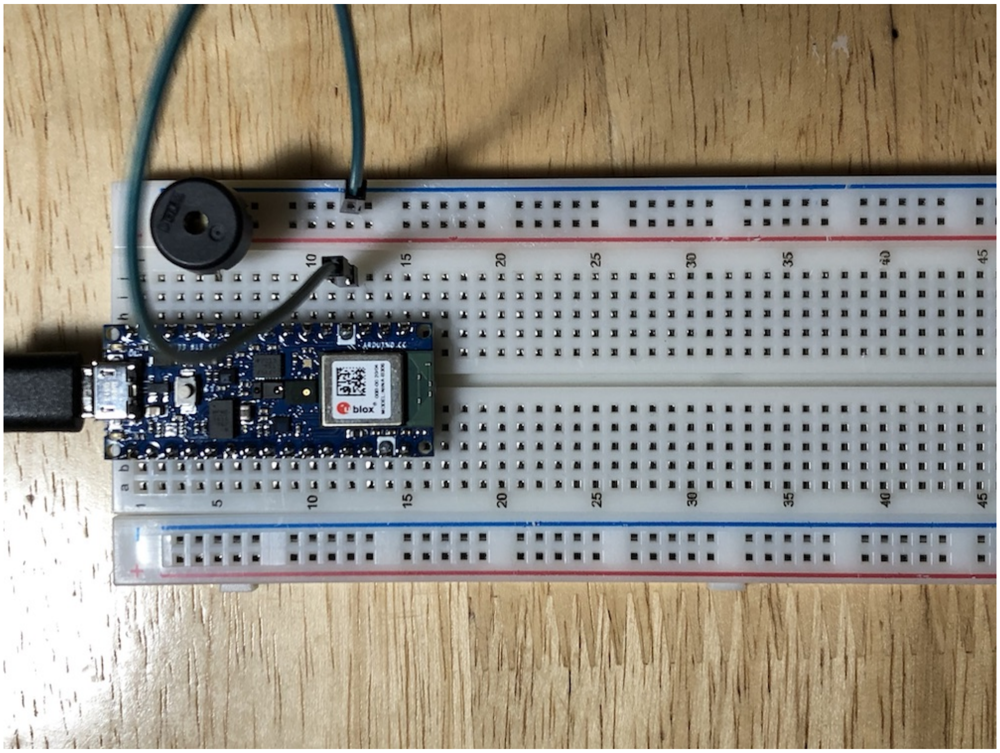
On the upper right corner of the window, click on the magnifying glass icon for opening the `Serial monitor`. Make sure the settings on the bottom match the ones on your computer, and that's it!



You uploaded your first example to your Arduino, which is now busy sending the messages you seen on the screen, and also showing all the different lights it has :)

Ground

Notice that the 14th pin on the left side and the 12th pin on the right side are labelled with white paint; this marks ground, also identified on the pinout. Take a wire (preferably green by convention for ground) and connect it from I12 to anywhere on the top righthand negative rail (the upper 25 pins), like this:



Outputs

Buzzer

Next, connect one of the legs of the piezo buzzer to the node labelled D8 on the pinout (which should be row 5 on the breadboard). Connect the other leg to the ground rail. Your wiring should look like this:

Installing the piezo buzzer

Now you're good to go! Upload `check_buzzer` to the microcontroller, open the serial monitor (top right button in the Arduino IDE), and follow the instructions from there!

LED

MIDI

MIDI Din jack

5 pins, only 3 are used.

Printer

We are using a thermal printer from Adafruit.

https://www.adafruit.com/product/2753

It has 5 cables:

VH - red - connect to the power supply 5V - 9V DTR - yellow - connect to GND on the Arduino TX - green - data out of the printer RX - blue - data in to the printer GND - black - connect to GND on the Arduino

We use a power supply, whose ground is connected to the one on the Arduino.

The power supply is 9V, center positive. Here is one available:

https://www.adafruit.com/product/276

Serial

Use a micro USB cable to connect to a computer.

Servo

The servo we are using has three cables:

- Yellow: signal
- Orange: power
- Brown: ground

Contributing

Issues

If you find an error or have a comment, please start a discussion by submitting an issue on our repositories!

- https://github.com/montoyamoraga/tiny-trainable-instruments/issues
- https://github.com/montoyamoraga/TinyTrainable/issues

Pull requests

Here is a step by step guide to make pull requests to this repository.

- Create a free GitHub account
- Fork the repository
- Clone your new repository to your computer

```
git clone https://github.com/your_username/tiny-trainable-instruments.git
```

Optionally, you can also clone the submodules of this repository, with the command

```
git submodule update --init --recursive
```

- Change directory (cd) into the project folder

```
cd tiny-trainable-instruments
```

- Make your changes

- Stage and make a commit to your repository on your computer

```
git add .
```

```
git commit -m "your comment"
```

- Push your commit to your personal fork on GitHub

```
git push
```

- Open your repository online
- Open ayour pull request and wait for comments or approval

Contributing documentation

For more information about how to contribute documentation to an open source artistic project, we recommend looking at the documentation by the p5.js project, available at https://github.com/processing/p5.js/blob/main/contributor_docs/contributing_documentation.md

Adding submodules

If you think there are more repositories we should include as submodules for archival purposes, use the following command, replacing GITPATH with the location of the repository you want to include, and FOLDERPATH with the destination.

```
git submodule add GITPATH FOLDERPATH
```

Helper scripts

The helper scripts are located on the assets/ folder. To run them, cd to assets/ and then use the following commands

Compiling code

This script uses arduino-cli for checking the compilation of all examples of TinyTrainable.

```
sh compile-code.sh
```

Delete metadata

This script uses exiftool to delete the metadata of all pictures in docs/

```
sh delete-metadata.sh
```

Format code

This script uses clang-format to organize all the code in TinyTrainable

```
sh format-code.sh
```

Markdown to PDF

This script uses pandoc to convert the documentation from Markdown to PDF format.

```
sh markdown-to-pdf.sh
```

Appendix D

Open source contributions

Nos debemos tanta plata

Los unos a los otros

Me dan escalofríos

Viernes

Tus Amigos Nuevos, 2013

During this thesis I contributed the following pull requests to open source projects that are either direct dependencies or inspirations.

Author	Repository	Contribution
Adafruit	[1, Adafruit_SSD1306]	[21, Format binary numbers]
tinyMLx	[28, TinyMLx Arduino Library]	[22, Update architecture name]
Yining Shi	[27, ML for Physical Computing]	[23, Fixed some typos]

Table D.1: Pull requests to open source projects

Appendix E

Rules of thumb

Wishes on a wheel
How it's supposed to feel

Wishes
Beach House, 2012

During this thesis I have tried to follow these rules of thumb:

- Openly share small steps
- Learn by failing often
- Contribute back
- Use tools you like
- Cite other people
- Sleep as much as possible

Bibliography

[1] Adafruit. Arduino library for ssd1306 monochrome 128x64 and 128x32 oleds. https://github.com/adafruit/Adafruit{_}SSD1306, 2021. [Online; accessed 31-July-2021].

[2] Adafruit. Breadboard-friendly midi jack (5-pin din): Id 1134. https://www.adafruit.com/product/1134, 2021. [Online; accessed 30-July-2021].

[3] Adafruit. Full sized breadboard: Id 239. https://www.adafruit.com/product/239, 2021. [Online; accessed 28-July-2021].

[4] Adafruit. Micro servo [ps1240]: Id 169. https://www.adafruit.com/product/169, 2021. [Online; accessed 30-July-2021].

[5] Adafruit. Mini thermal receipt printer starter pack: Id 600. https://www.adafruit.com/product/600, 2021. [Online; accessed 30-July-2021].

[6] Adafruit. Monochrome 0.91" 128x32 i2c oled display - stemma qt / qwiic: Id 4440. https://www.adafruit.com/product/4440, 2021. [Online; accessed 30-July-2021].

[7] Adafruit. Piezo buzzer [ps1240]: Id 160. https://www.adafruit.com/product/160, 2021. [Online; accessed 30-July-2021].

[8] Adafruit. Premium male/male jumper wires - 40 x 6" (150mm): Id 758. https://www.adafruit.com/product/758, 2021. [Online; accessed 28-July-2021].

[9] Adafruit. Super bright white 5mm led (25 pack): Id 754. https://www.adafruit.com/product/754, 2021. [Online; accessed 30-July-2021].

[10] Adafruit. Usb cable - usb a to micro-b - 3 foot long: Id 592. https://www.adafruit.com/product/592, 2021. [Online; accessed 28-July-2021].

[11] Arduino. Arduino nano 33 ble sense with headers | arduino official store. https://store.arduino.cc/usa/nano-33-ble-sense-with-headers, 2021. [Online; accessed 28-July-2021].

[12] Arduino. Arduino nano 33 iot with headers | arduino official store. https://store.arduino.cc/usa/nano-33-iot-with-headers, 2021. [Online; accessed 01-August-2021].

[13] Arduino. Arduino uno rev3 | arduino official store. https://store.arduino.cc/usa/arduino-uno-rev3, 2021. [Online; accessed 31-July-2021].

[14] Aarón Montoya-Moraga Gaurav Patekar. Open drawing machine. https://github.com/montoyamoraga/open-drawing-machine, 2021. [Online; accessed 01-August-2021].

[15] Critter & Guitari. Kaleidoloop. https://web.archive.org/web/20150206042159/http://www.critterandguitari.com/collections/instruments/products/kaleidoloop, 2015. [Online; accessed 24-July-2021].

[16] Critter & Guitari. Official website. https://www.critterandguitari.com/, 2021. [Online; accessed 24-July-2021].

[17] Bastl Instruments. Official website. https://www.bastl-instruments.com/, 2021. [Online; accessed 24-July-2021].

[18] Sam Lavigne. Training poses. https://lav.io/projects/training-poses/, 2018. [Online; accessed 26-July-2021].

[19] Monome. Official website. https://www.monome.org/, 2021. [Online; accessed 24-July-2021].

[20] Aarón Montoya-Moraga. Introduction to computer networks for artists. https://github.com/montoyamoraga/intro-to-computer-networks-for-artists, 2021. [Online; accessed 01-August-2021].

[21] montoyamoraga. change format of binary numbers. https://github.com/adafruit/Adafruit{_}SSD1306/pull/209, 2021. [Online; accessed 31-July-2021].

[22] montoyamoraga. update name of architecture. https://github.com/tinyMLx/arduino-library/pull/8, 2021. [Online; accessed 31-July-2021].

[23] montoyamoraga. Update readme.md. https://github.com/yining1023/Machine-Learning-for-Physical-Computing/pull/1, 2021. [Online; accessed 31-July-2021].

[24] Trevor Pinch and Frank Trocco. *Analog days: the Invention and impact Of The Moog synthesizer*, chapter 3: Shaping the Synthesizer, page 68. Harvard University Press, first harvard university press paperback edition edition, 2004.

[25] PJRC. Teensy-lc usb development board. https://www.pjrc.com/store/teensylc_pins.html, 2021. [Online; accessed 31-July-2021].

[26] Shbobo. Official website, 2021. [Online; accessed 24-July-2021].

[27] Yining Shi. Introduction to machine learning for physical computing. https://github.com/yining1023/Machine-Learning-for-Physical-Computing, 2021. [Online; accessed 31-July-2021].

[28] tinyMLx. Harvard_tinymlx arduino library. https://github.com/tinyMLx/arduino-library/, 2021. [Online; accessed 31-July-2021].