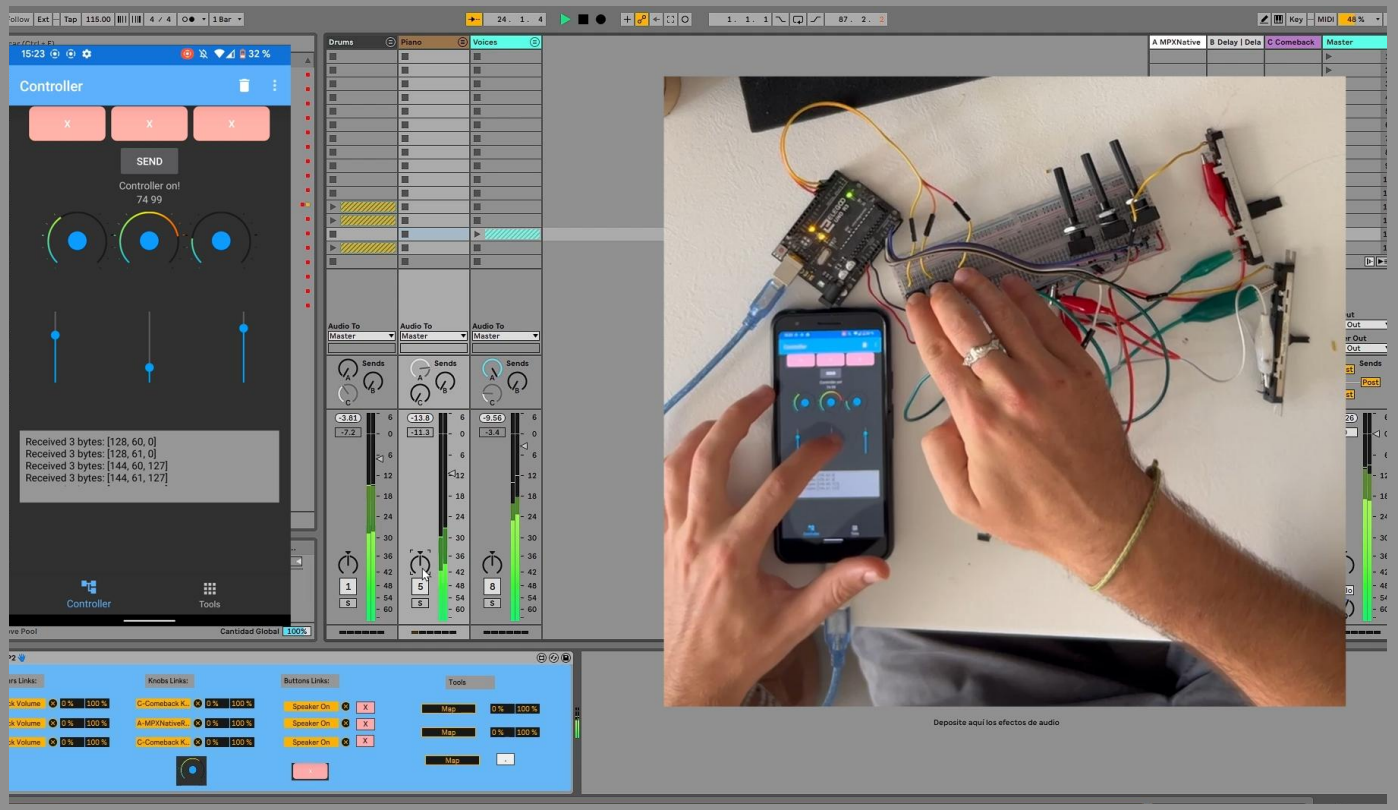
Bachelor's Thesis
[13349]

Reuse of a Mobile Phone as a MIDI Controller



Lucas Fuentes-Cantillana Monereo (100429218)

Bachelor's Degree in Sound and Image Engineering

Tutor: Carlos Rodriguez Mayor

Date:
Jul, 8, 2025

Table of Contents

Introduction

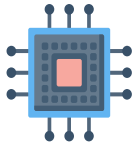
Bachelor's Thesis

[13349]

Context and Motivation

Reuse of an electronic project

- Get a second use of Ics
- Reuse a sensor by reverse engineering
- Reuse screens
- Reuse DACs and ADCs
- etc.



Reuse at the primary level

- Image Analysis → Condition of Components
- Recycle components (RLC, diodes, transistors, etc.)
- Study of electrical components
- Recycle batteries

- **Virtual Circuit Creation**
  - Virtual Analog Systems (VAS)

Second uses for mobile phones (simple)

- Security Camera
- Sensor for farms or land
- Customizable Alarms

Second uses

Jungle Sounds Guard(Topher White)

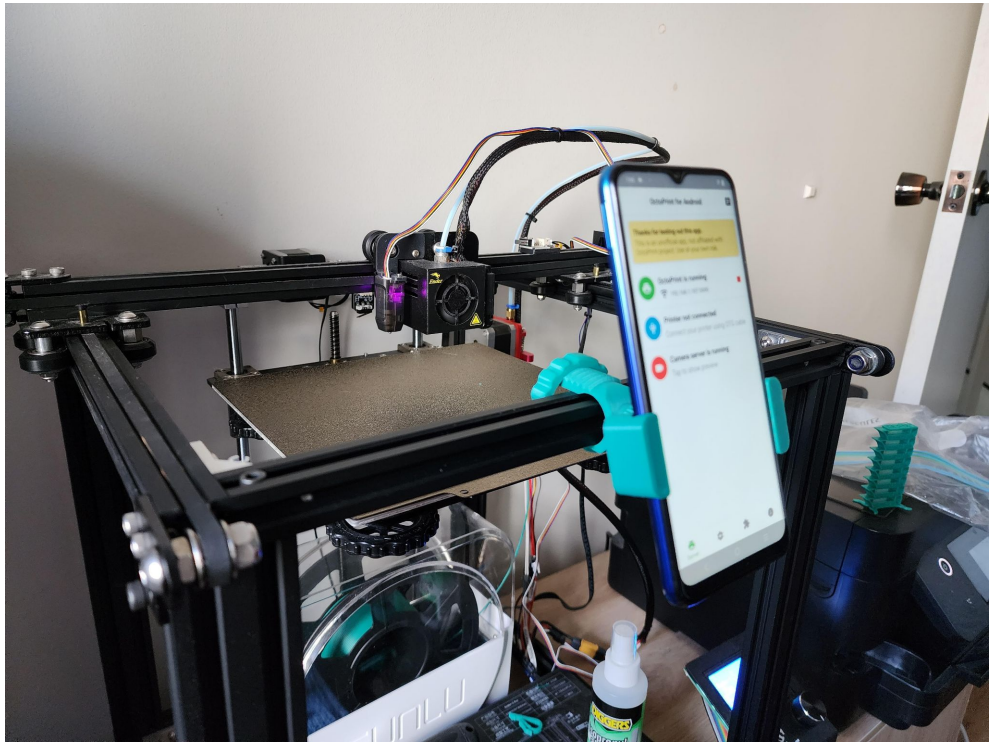
- Jungle Sound Receivers:
- **Identify sounds** chainsaws
- **Guards notified** remote from their position

Electronic Ears to listen to the forest

NATIONAL GEOGRAPHIC

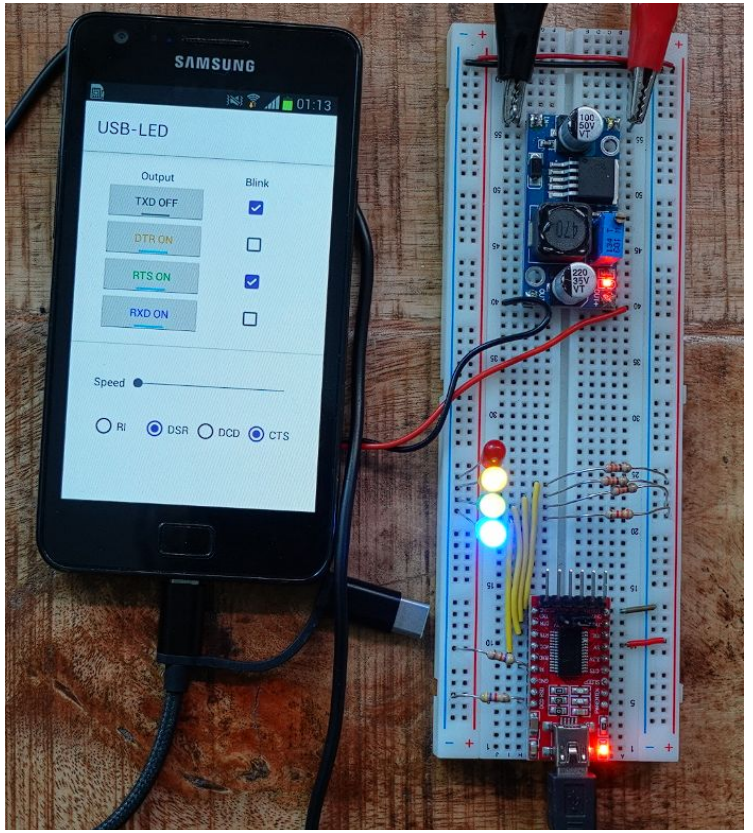
Second uses

- **Remote controller for 3D machines (Octo4a)**



Octo4a

- **Communication with UART from Android**

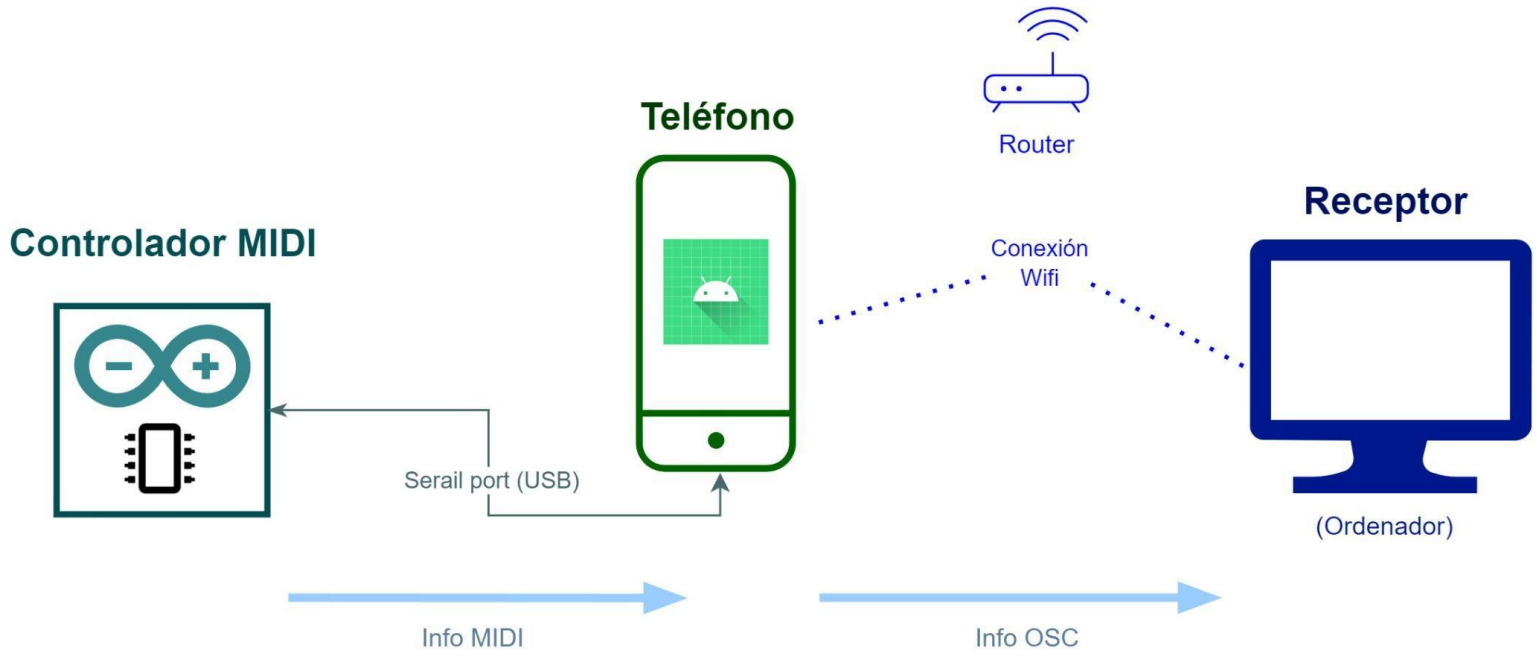


connection UART

Proyect

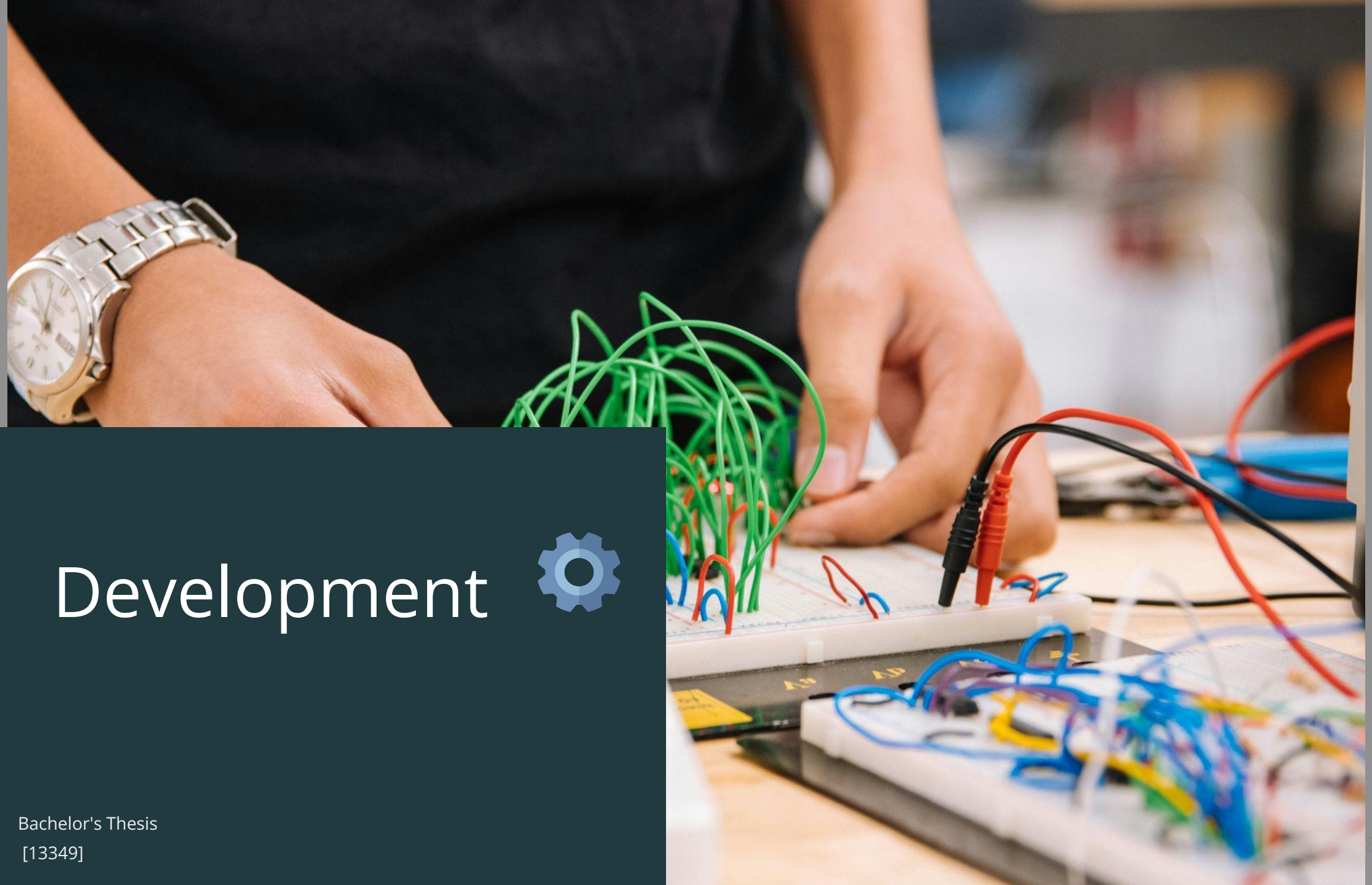
Idea

Use the mobile device as an intelligent interconnect to provide MIDI devices with wireless connection.



Connections

- **Controller – Phone**
  - Through Port Series
- **Phone - Computer:**
  - Through Wi-Fi connection(UDP)



Development

Bachelor's Thesis

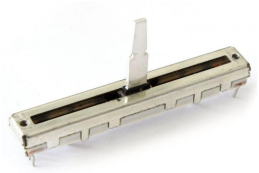
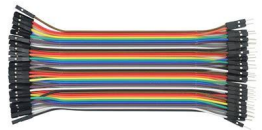
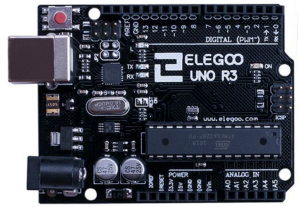
[13349]

Scenario

Controller

- 3x Faders
- 3x Knobs
- 3x Switches
- + Cables
- Arduino Uno
- Cable serial Port (USB B)



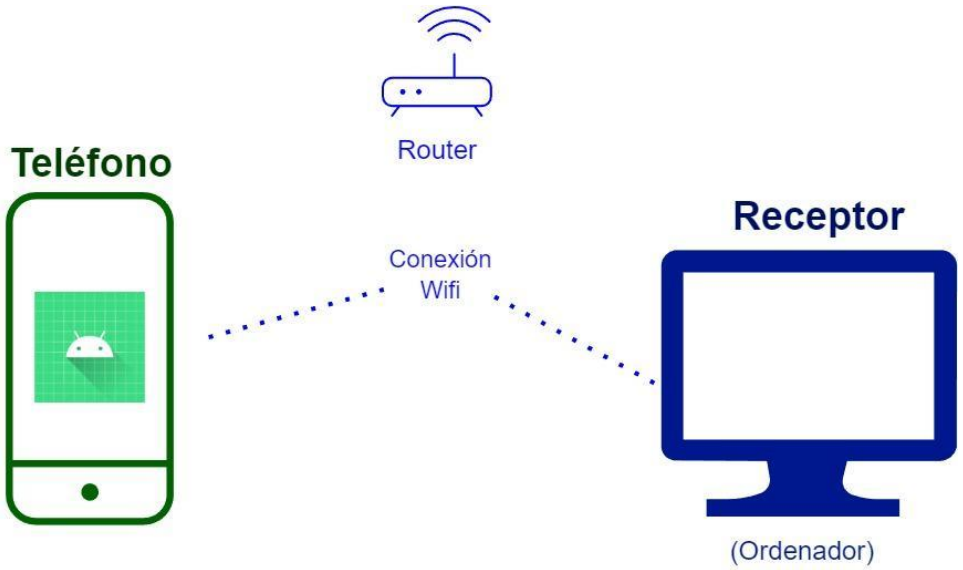




Sender (Mobile App)

- App developed in Android Studio (Pixel 3a)

Receiver (Computer Plugin)

- Plugin created in Max for Live for the Ableton Live program (Asus Zenbook)



Scenario

Controller

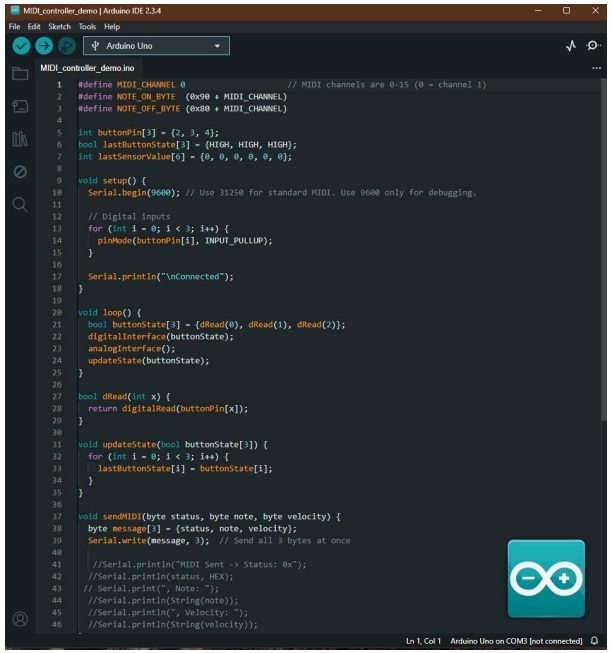
- Arduino IDE (C++)

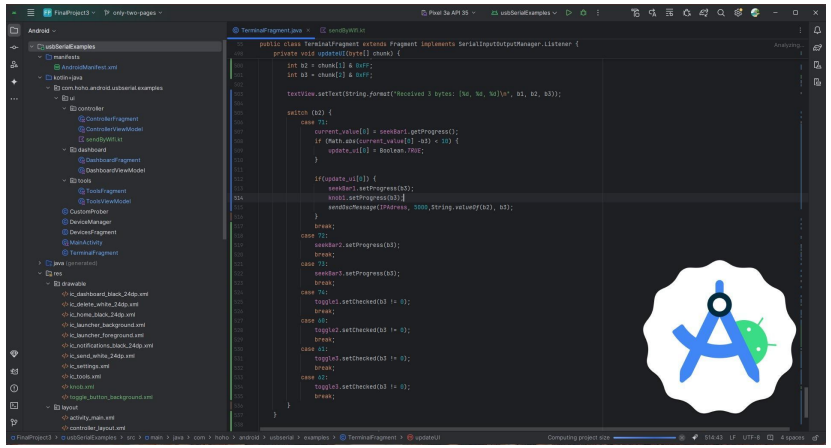
Sender (Mobile App)

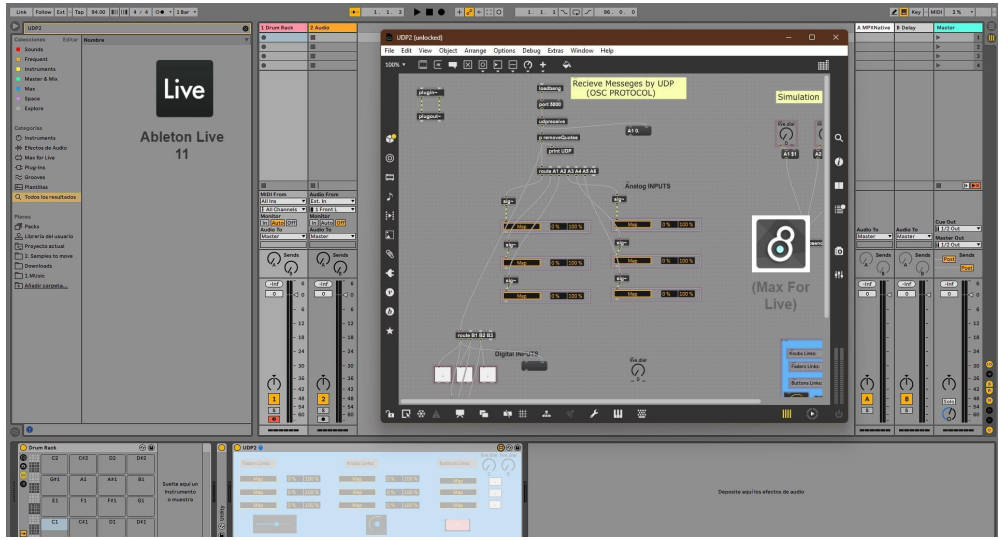
- Android Studio (Java, Kotlin)

Receiver (Computer Plugin)

- Plugin created in Max for Live for the Ableton Live program (nodes)







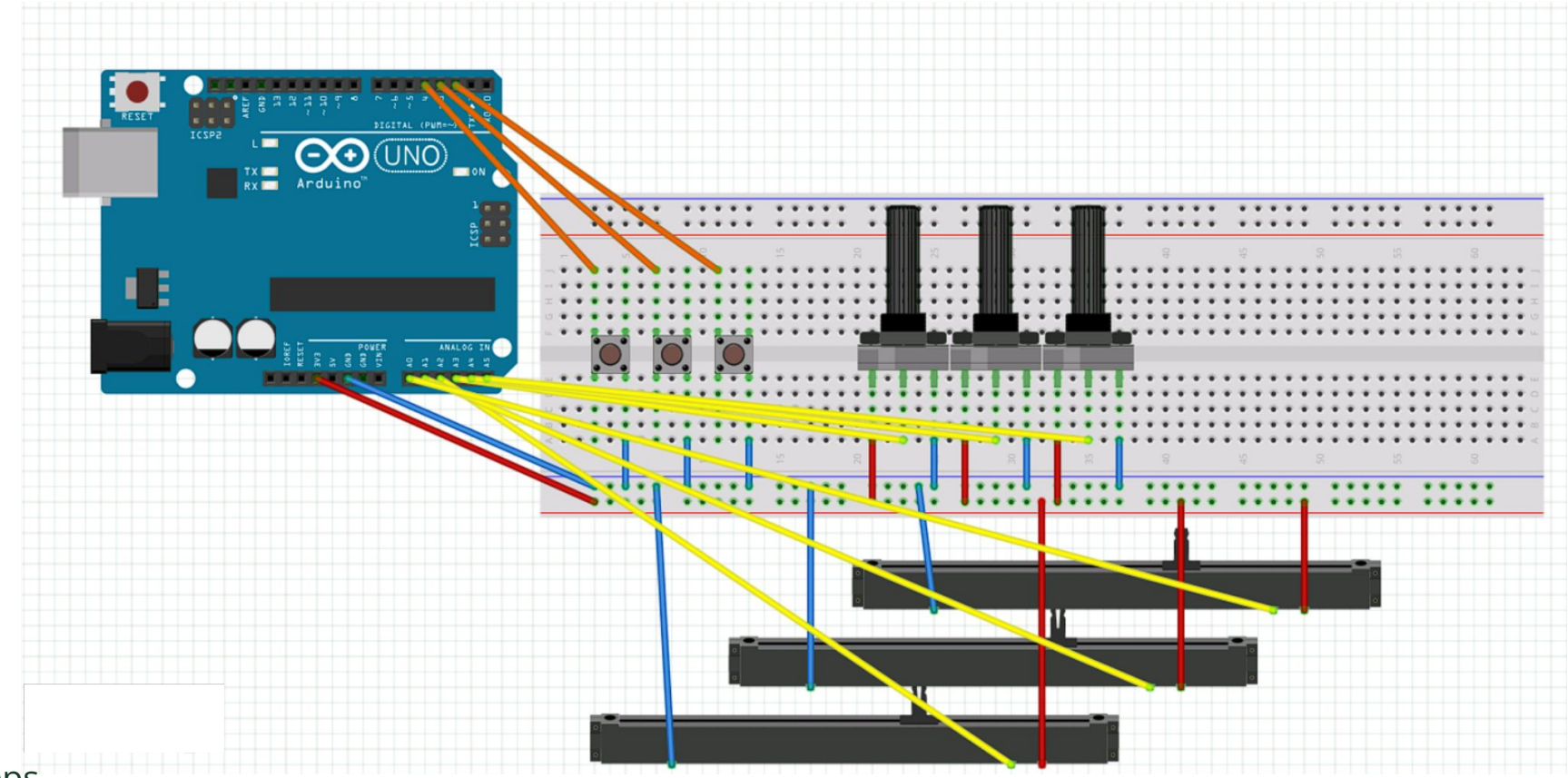
Circuit

Inputs:

- 3 Digital Inputs→
  - 3x switches
- 3 Analogic Entry
  - 3x Sliding
  - 3x Knobs

Configurations:

- Digital (pull up): On off
- Analogue →
  - MIDI has 128 values
  - Input Resolution→ 1048 steps



$$\text{Valor ADC} = \frac{V_{\text{entrada}}}{V_{\text{ref}}} \times \text{Resolución}$$

$$\text{Valor ADC} = \frac{3{,}3\,\text{V}}{5\,\text{V}} \times 1024 = 675{,}84$$

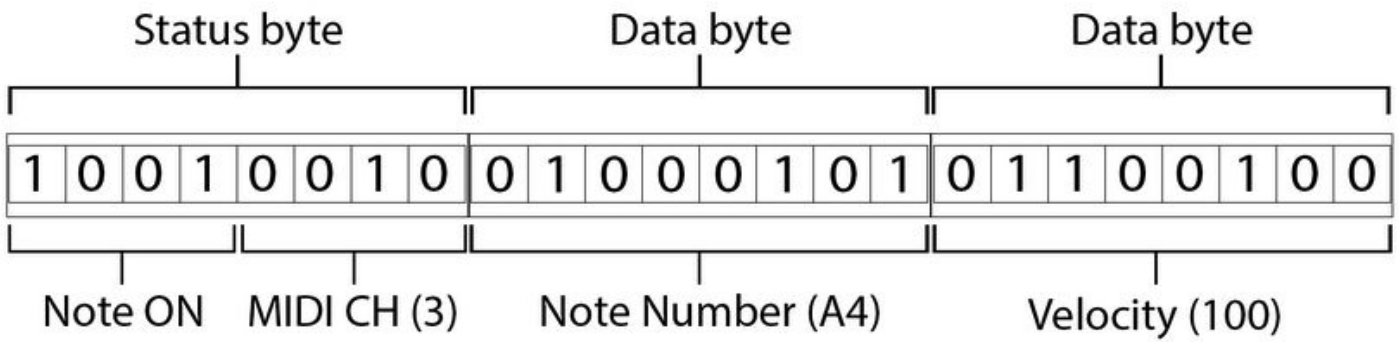
$$\text{Valor}(0, 127) = \frac{\text{Vanalog}}{675{,}84} \times 127$$

MIDI protocol

Format

- MIDI → Musical Instrument Digital Interface is a standard communication protocol that allows electronic musical instruments (1983)
- Streaming of music events.
- 3-byte packets (Status/Channel, Note and Value).

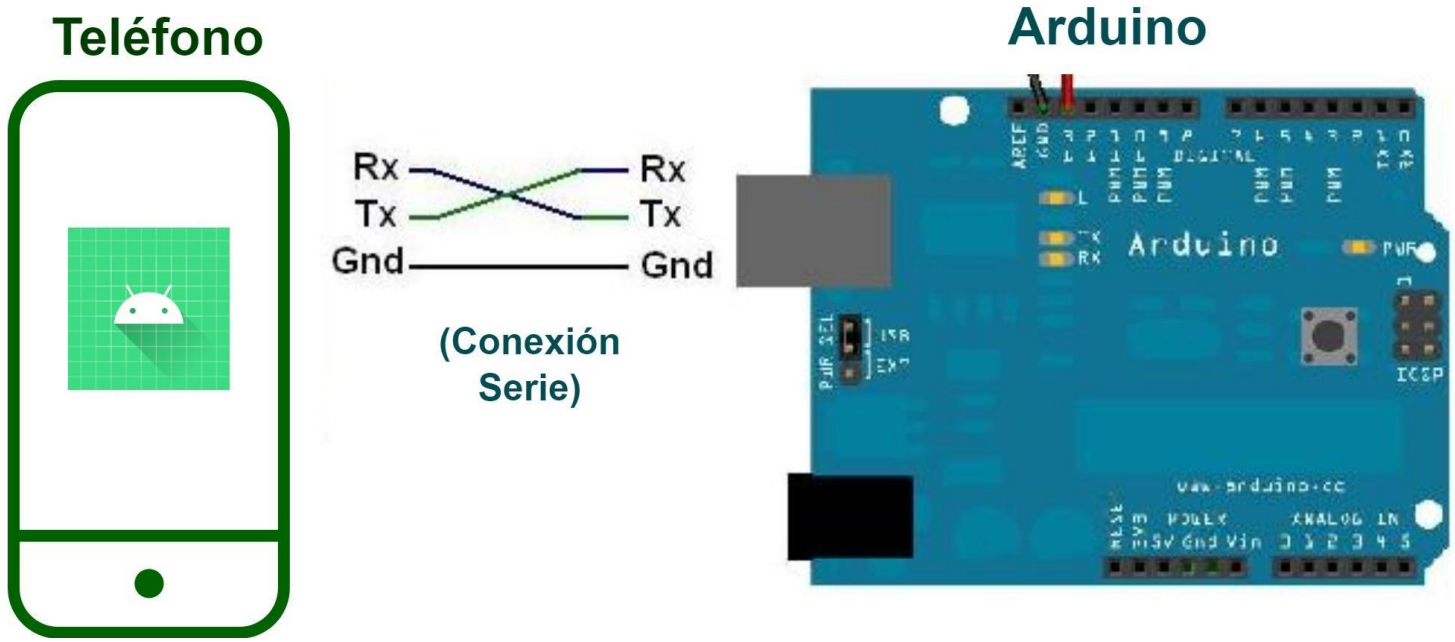
```
37  void sendMIDI(byte status, byte note, byte velocity) {
38    byte message[3] = {status, note, velocity};
39    Serial.write(message, 3);
40
41  }
```



Mobile Connection

Cable transmission:

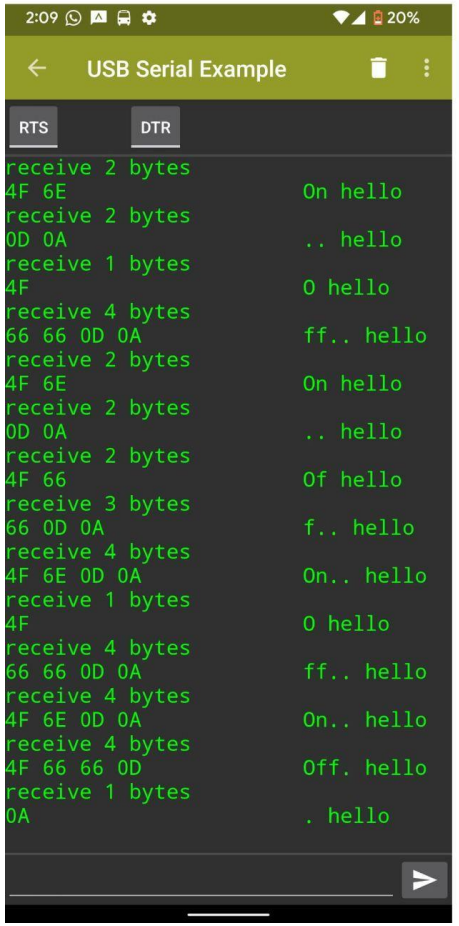
- Communication is used through the Serial Port or Serial Connection
- Cable OF USB - C → USB - B
- Baud rate : 96000 bits/s

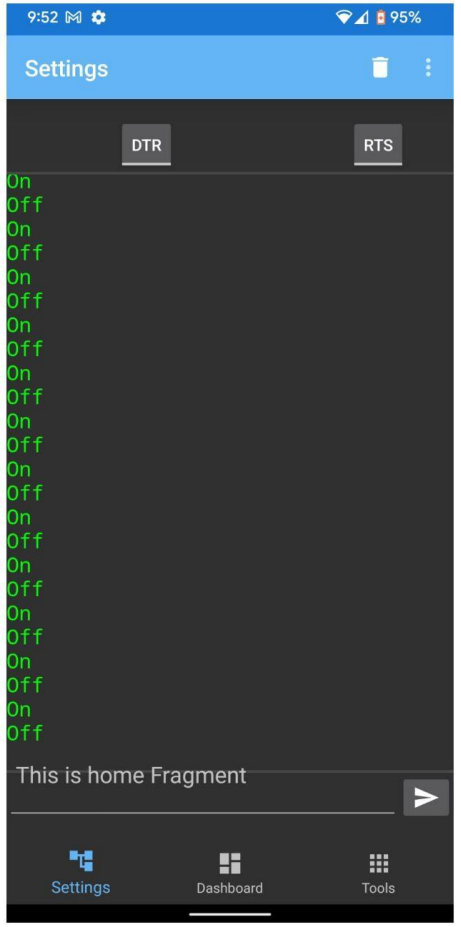


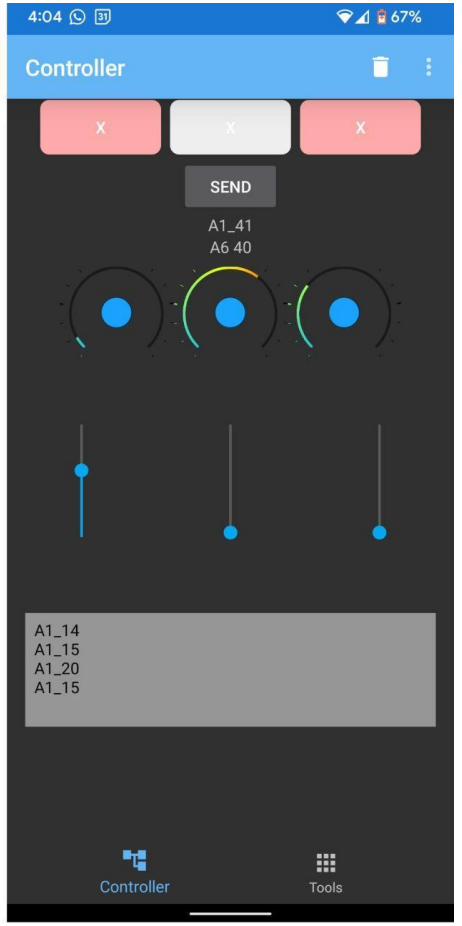
Evolution

Application Evolution

- The project began thanks to Kai Morich and his library (usb-serial -for -android)
- Interface changes to make it easier to use
  - 2 Navigation Buttons
  - Sort incoming messages
  - Enter the parameters





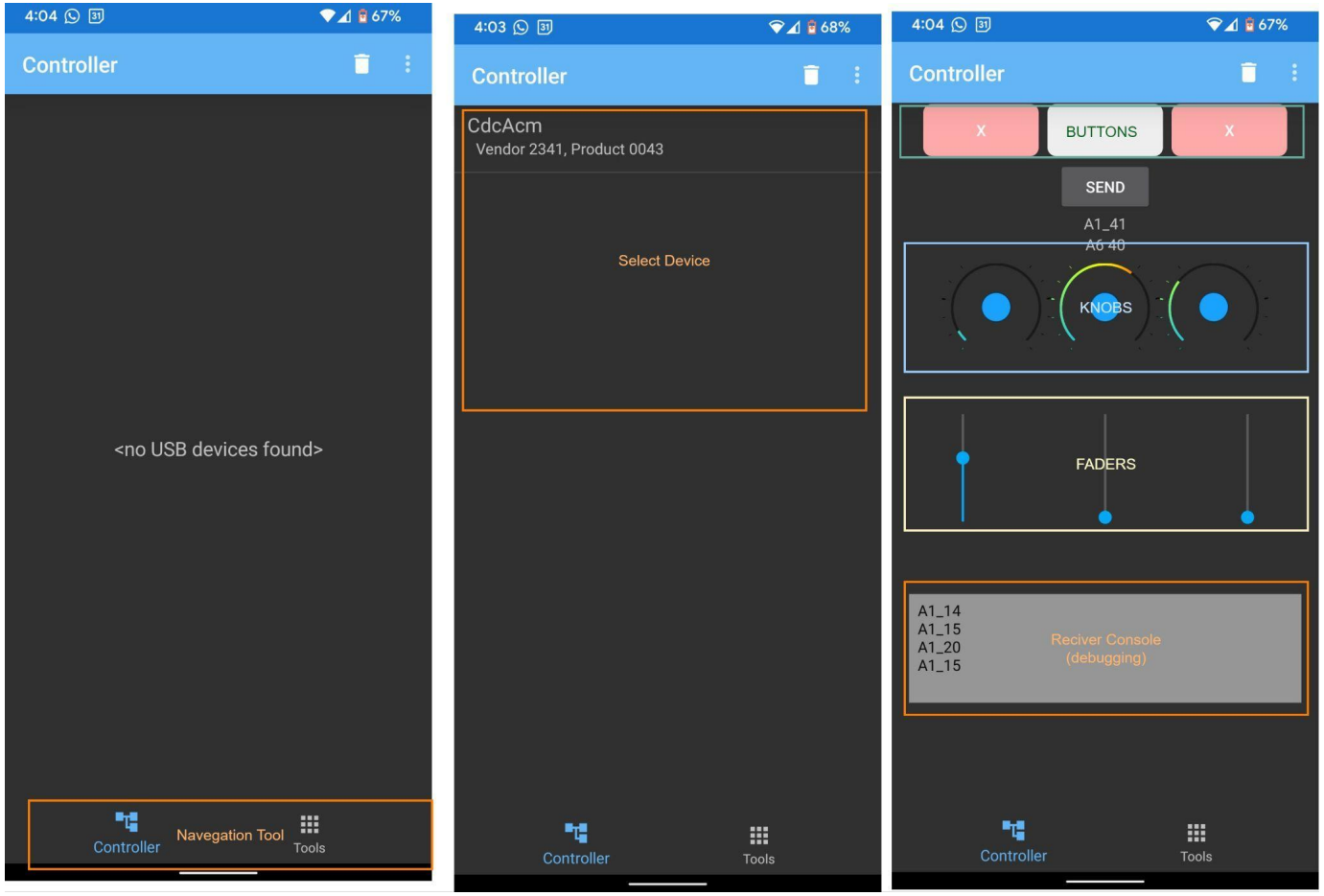


Interface

Layout:

1. **Identify connected controllers**
2. **Select the driver (in this case CdcAcm (driver identifier**
3. Interactive Interface:

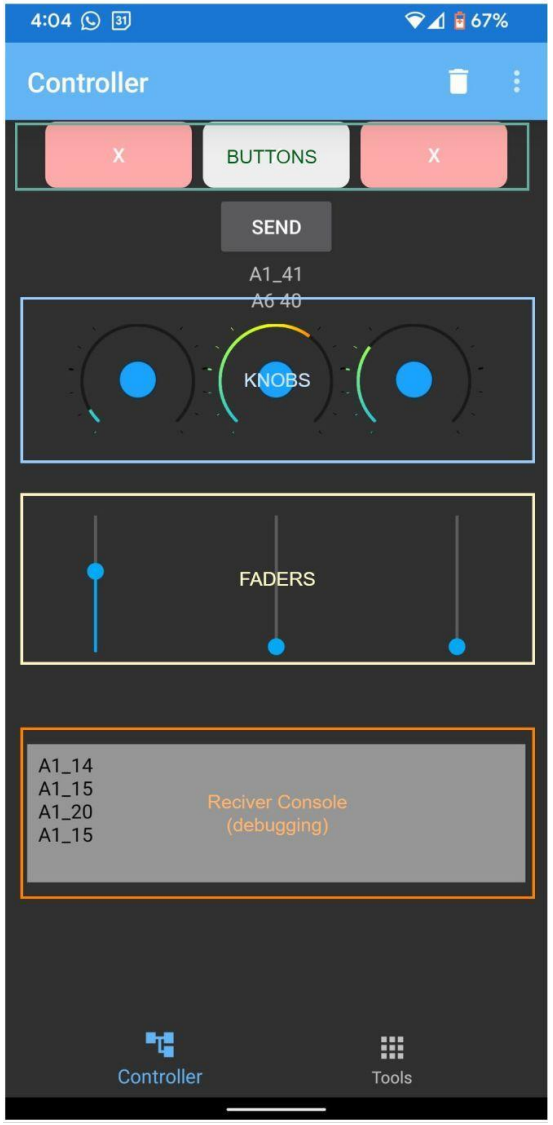
- The interface contains the same layout as the physical version.
- **Message terminal (debugging)**



Interface

Page Controller:

- **Switches** (x3)
- **Knobs** (x3)
  Made through the bookshop:[rotary-buttons]
- **Faders** (3x)
  Sliders o seekBar inside de Android Studio
- **Debugging terminal:**
  Get information received by Serial-Port



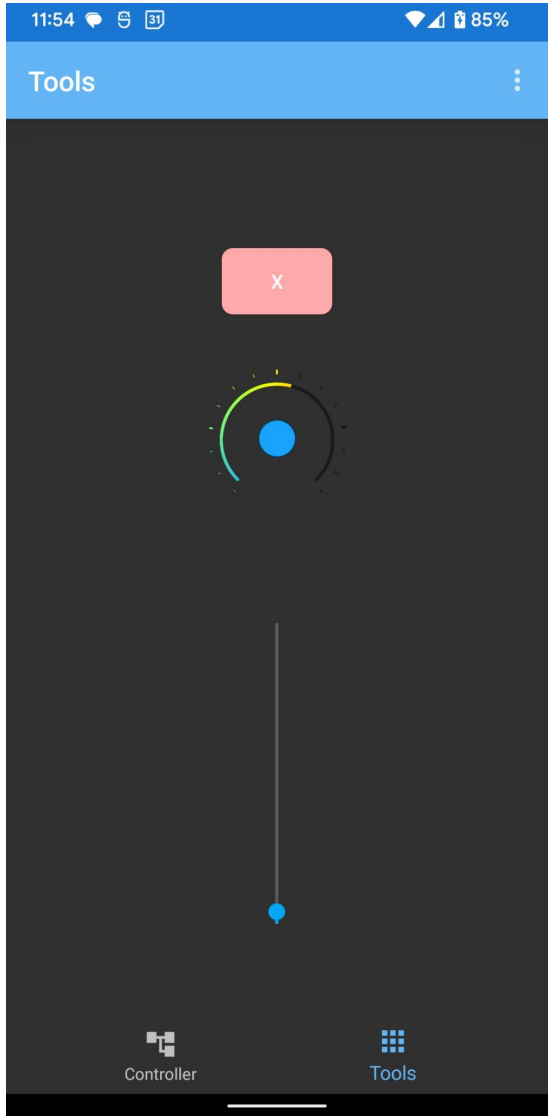
Interface

Page “Tools”:

1. Interactive single-channel interface (demo mode)

- It offers the same functionality as the controllers available on the "Controller" page.

- **Trial version. The idea is to increase the number of controllers and make it customizable.**



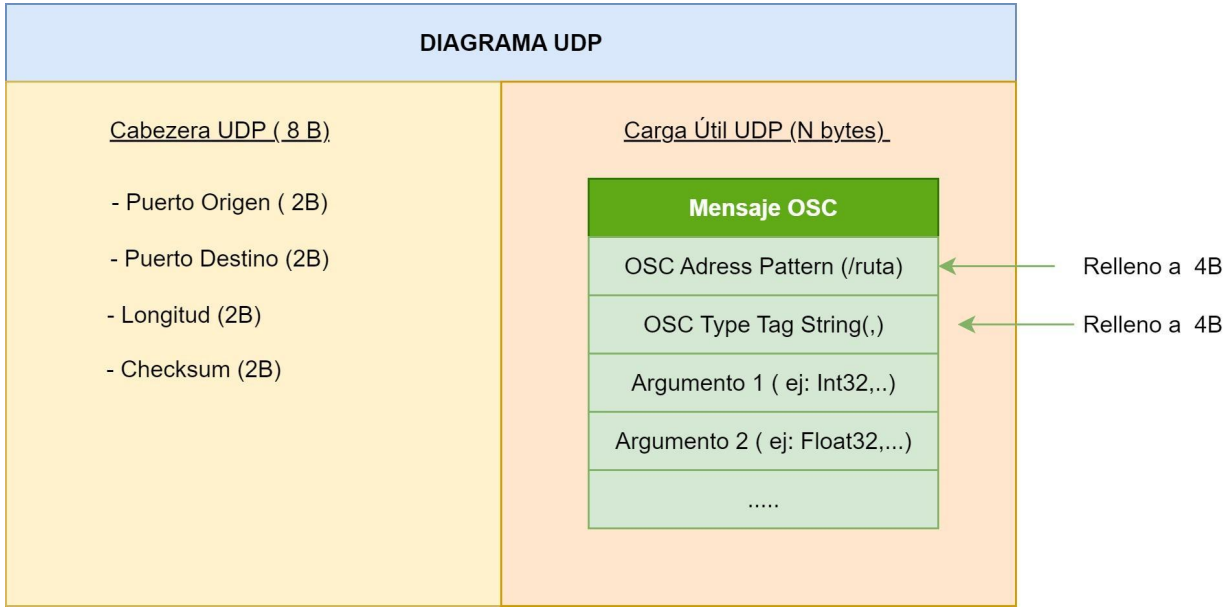
Shipping over the Wi-Fi network

Protocol UDP

- UDP (User Datagram Protocol) is a network transport protocol.
- **Fast, Lightweight and unreliable → not guaranteed reception.**

Contains:

- Route header (basic information for shipping).
- Payload, which in this case is an encapsulated OSC message.



Protocol OSC

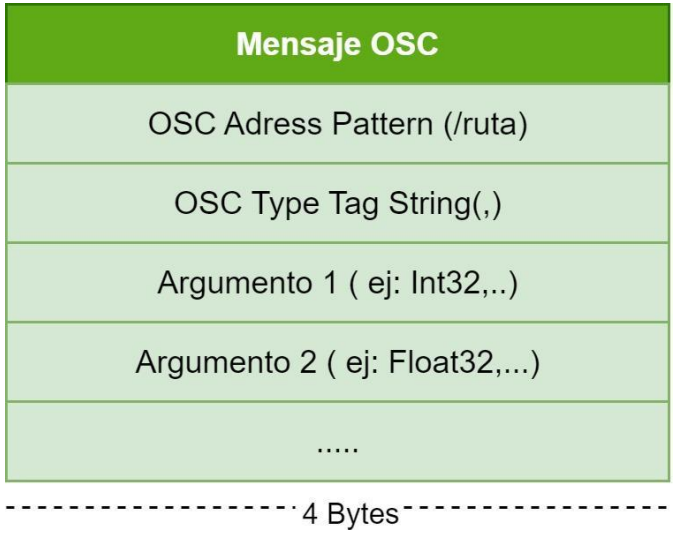
OSC Message Protocol:

- OSC = Open Sound Control → network protocol designed for communication between computers, synthesizers, controllers

- **Modern alternative to protocolMIDI →**
  - Supports modern data types (Strings , Float32, etc)

Contains:

- Route Header
- OSC Type → Tag in String format
- Stacked arguments in 4-byte format



Packaging

Protocol UDP y OSC

- Package messages in multiples of 4 Bytes
- Fill with Null Packets(o 0) the remaining gaps→ *padding*
- Use Socket Library to Send UDP Packets over the Local Network

```
fun sendOscMessage(serverIP: String, port: Int, address: String, data: Float) {
    Thread {
        val socket = DatagramSocket()

        val addressBytes = (address + "\u0000").toByteArray()
        val addressPaddedLength = (addressBytes.size + 3) / 4 * 4
        val paddedAddressBytes = ByteArray(addressPaddedLength)
        System.arraycopy(addressBytes, 0, paddedAddressBytes, 0, addressBytes.size)

        val typeTag = ",s\u0000\u0000".toByteArray()

        val messageBytes = (message + "\u0000").toByteArray()
        val messagePaddedLength = (messageBytes.size + 3) / 4 * 4
        val paddedMessageBytes = ByteArray(messagePaddedLength)
        System.arraycopy(messageBytes, 0, paddedMessageBytes, 0, messageBytes.size)

        val packetBytes = ByteArray( size: paddedAddressBytes.size + typeTag.size + paddedMessageBytes.siz
        var pos = 0
        System.arraycopy(paddedAddressBytes, 0, packetBytes, pos, paddedAddressBytes.size)
        pos += paddedAddressBytes.size
        System.arraycopy(typeTag, 0, packetBytes, pos, typeTag.size)
        pos += typeTag.size
        System.arraycopy(paddedMessageBytes, 0, packetBytes, pos, paddedMessageBytes.size)
```

Local Network Connection

Device identification

- The target device is searched on the local network using its Hostname:
  - Example: (ASUS-ZENBOOK)

Found:

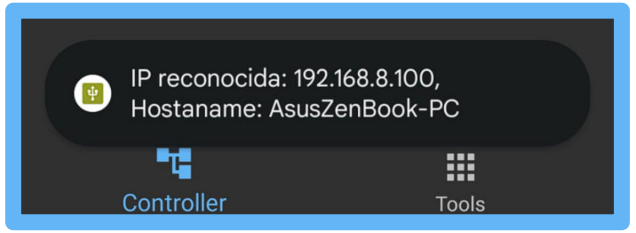
- Display a Toast message→
  ➤ IP acknowledged→ IP +Number

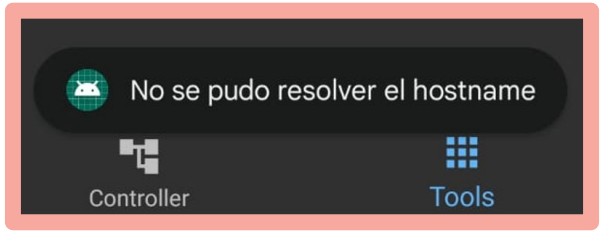
Not Found:

- Display a Toast message→
  ➤ Unable to resolve hostname

```
private void obtainIP(Context context) {
    new Thread(() -> {
        try {
            String hostname = "AsusZenBook-PC";
            InetAddress address = InetAddress.getByName( host: hostname+".local");
            IPAddress= address.getHostAddress();
            Log.d( tag: "HOSTNAME",  msg: "IP: " + IPAddress);

            new Handler(Looper.getMainLooper()).post(() ->
                    Toast.makeText(context,  text: "IP reconocida: " + IPAddress
                            + ", Hostaname: "+ hostname, Toast.LENGTH_LONG).show()
            );
```



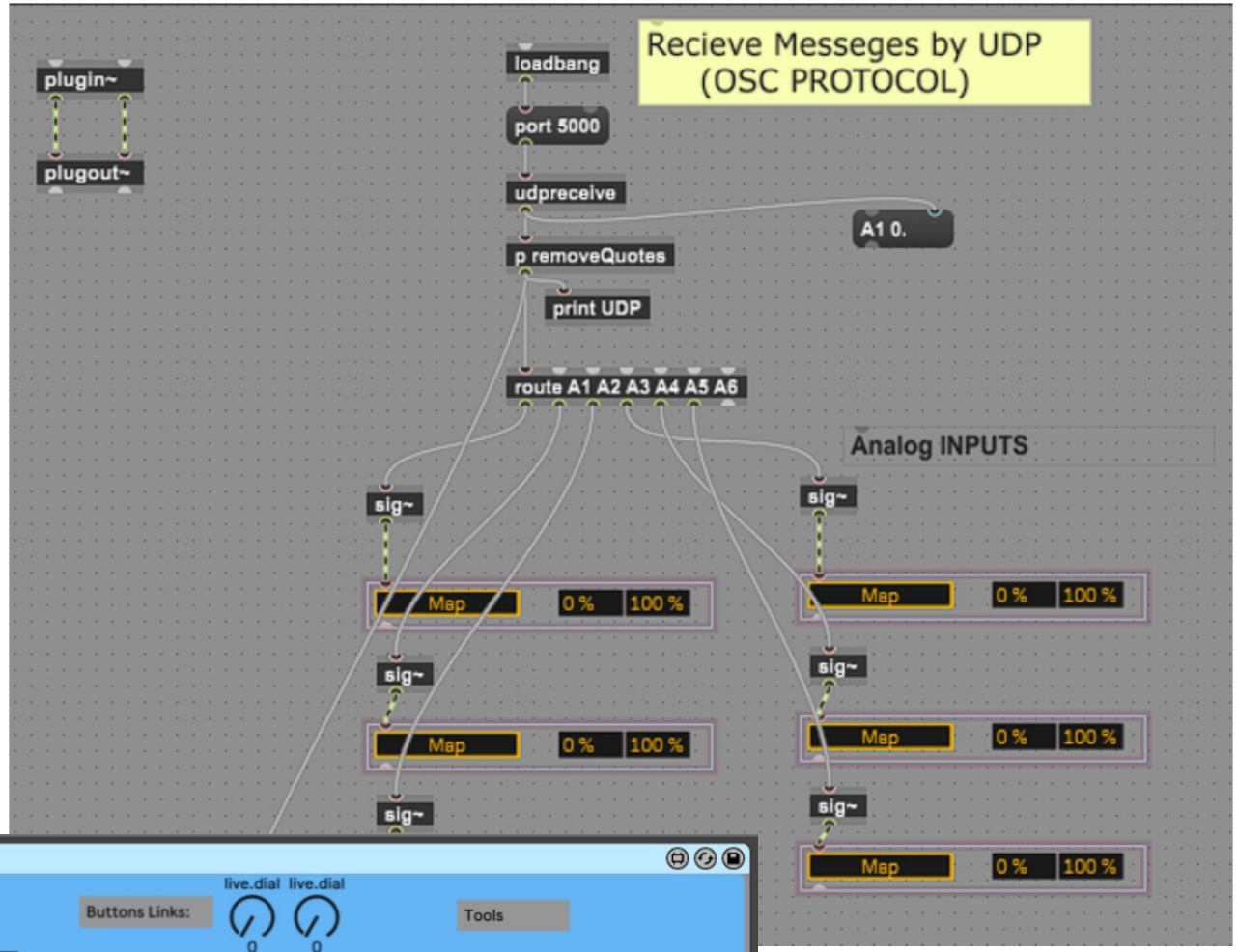


Plugin

Max for Live plugin:

Max for Live is a node-based visual programming environment, integrated into Ableton Live.

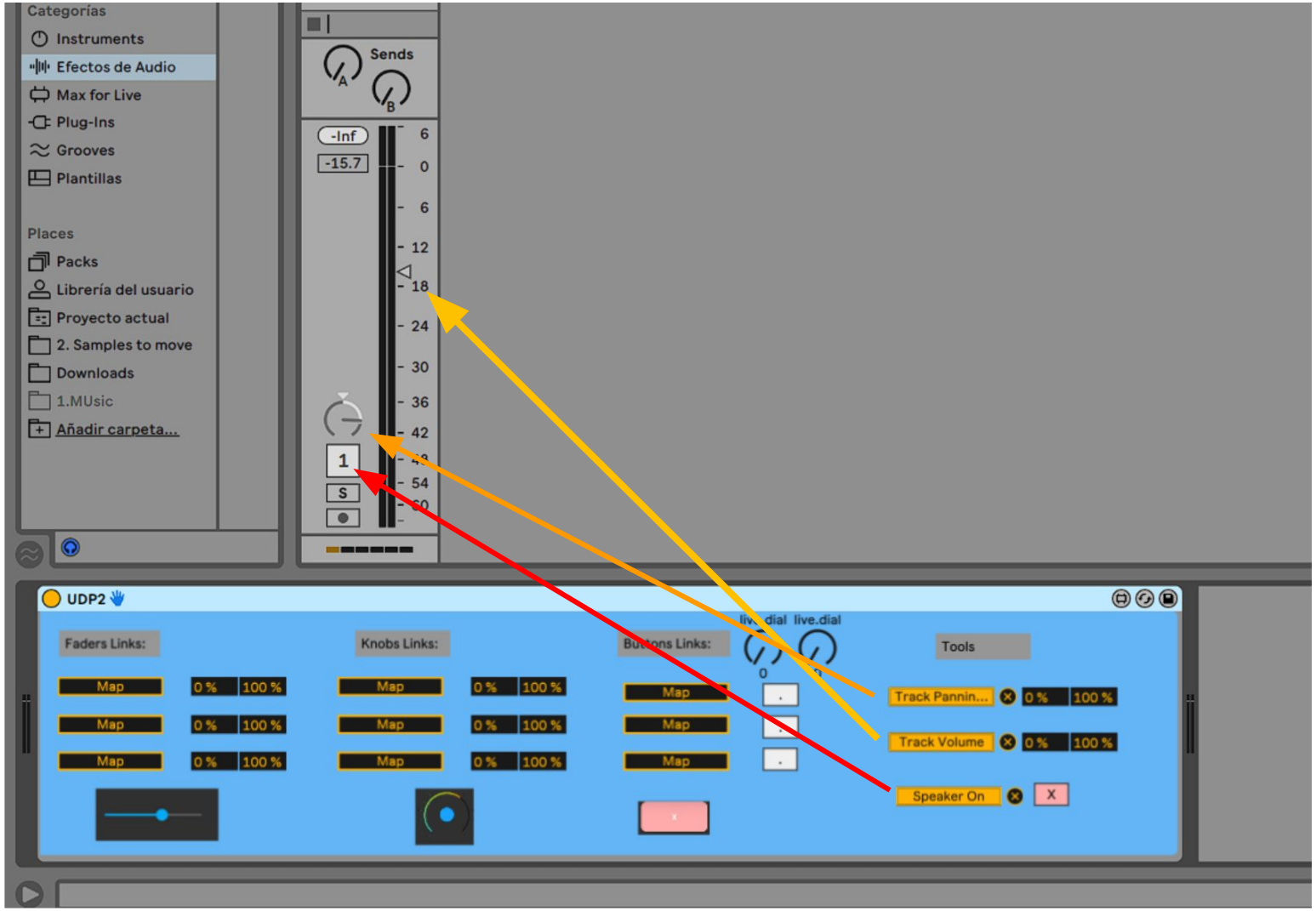
- The plugin receives messages via UDP using the OSC protocol
- Depending on its identifier (for example: "A1"), it redirects messages to the corresponding parameter



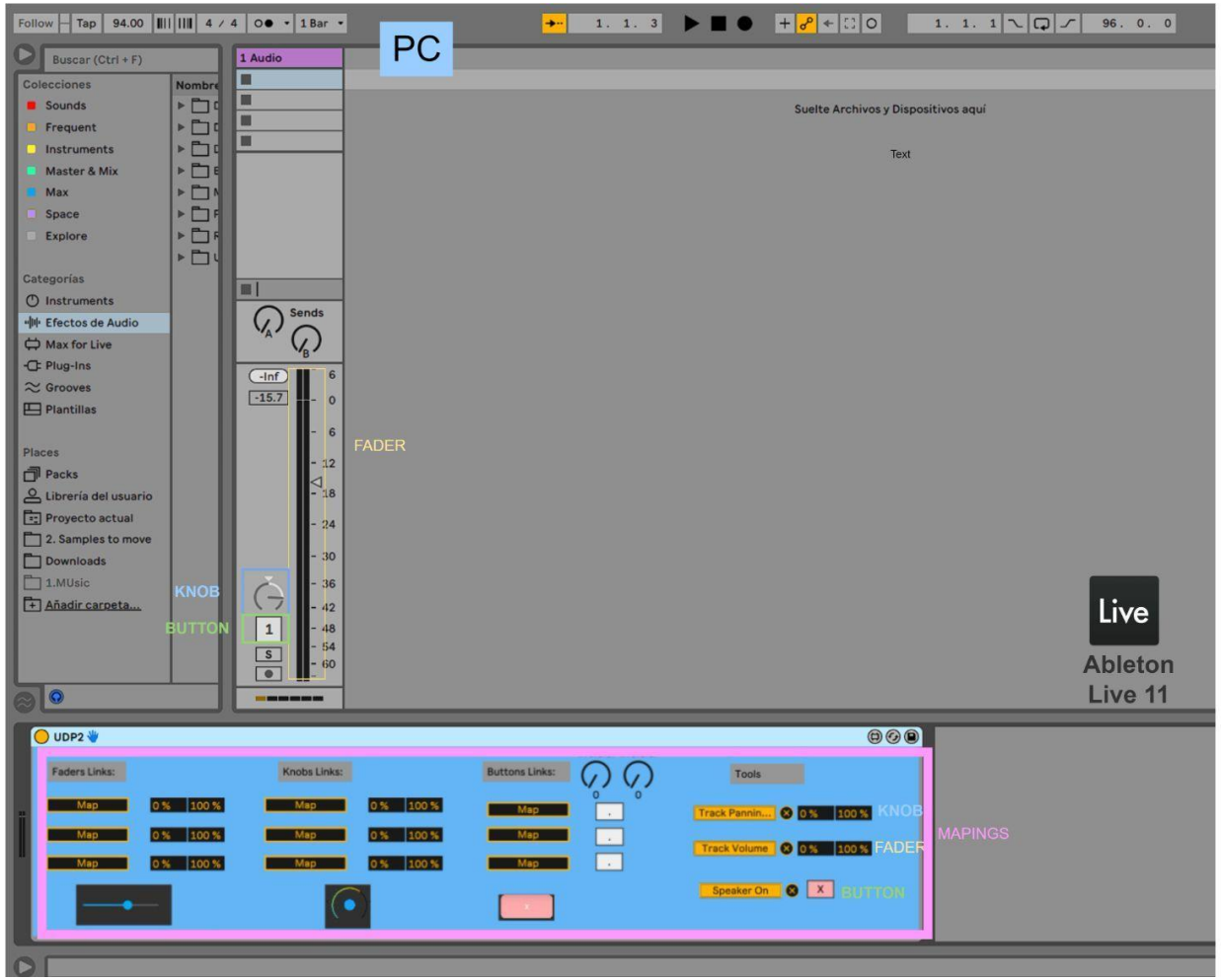
Plugin

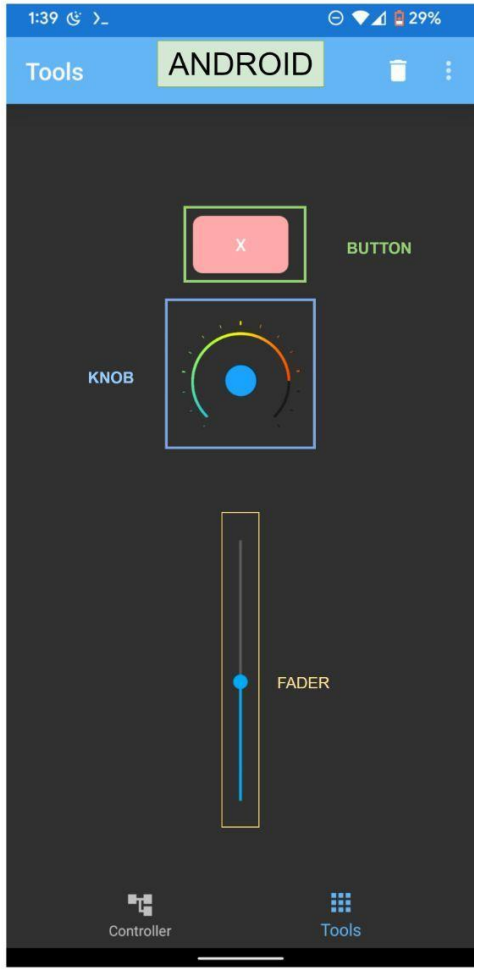
Ableton Live plugin

- The interface is used to map, control parameters within the program.
- Control any program parameter
- Real-time control (latency 2-60 ms)



Grouped


PC
Suelte Archivos y Dispositivos aquí
Text
FADER
KNOB
BUTTON
Live
Ableton
Live 11
UDP2
Faders Links:
Knobs Links:
Buttons Links:
Tools
Track Pannin...
Track Volume
Speaker On
KNOB
FADER
BUTTON
MAPINGS

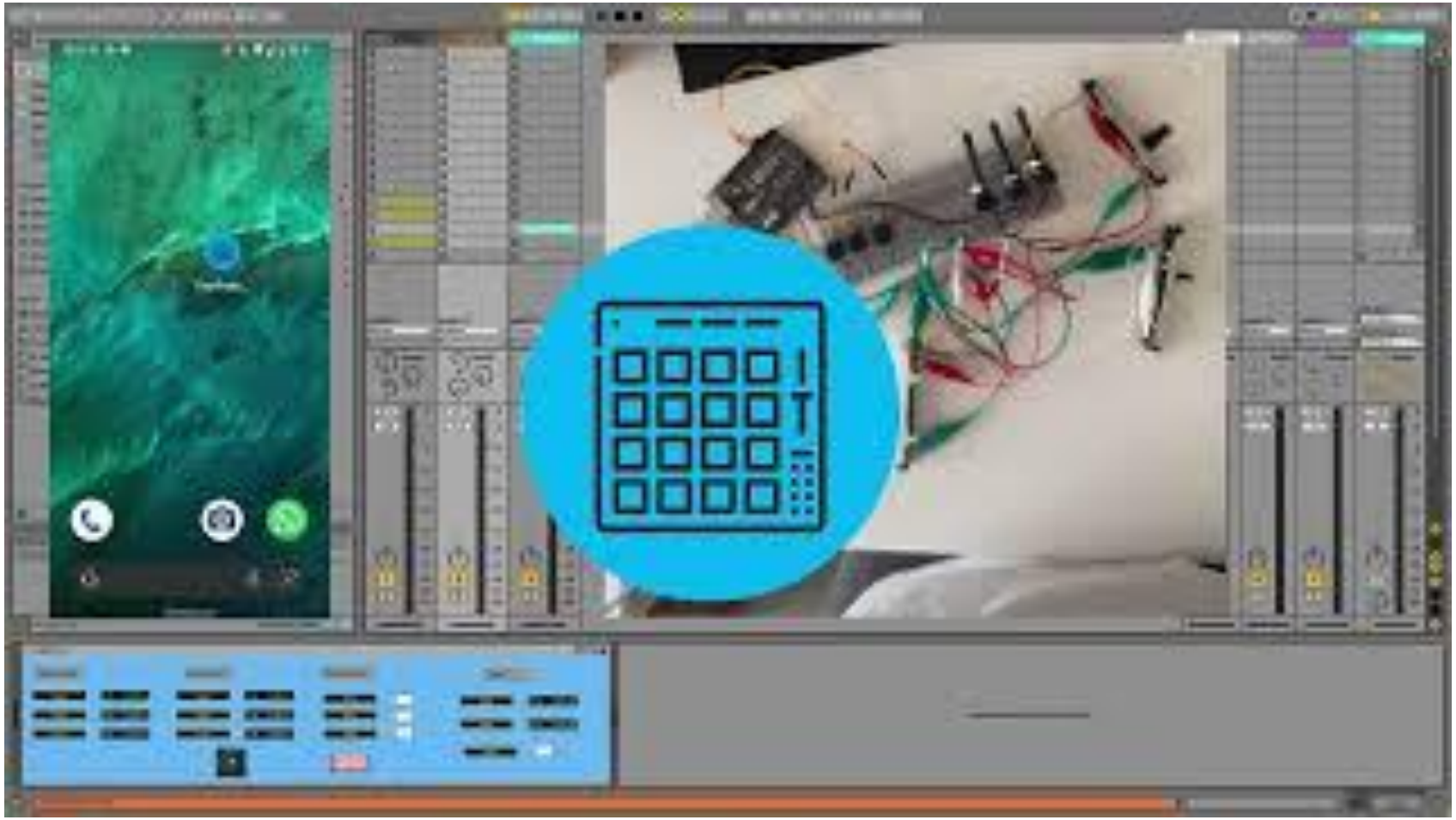

Tools
ANDROID
BUTTON
KNOB
FADER
Controller
Tools

Conclusions

Bachelor's Thesis

[13349]

Project Demo



Latency

Network Latency Calculations

- UTC Arrival Time - UTC Departure Time.
- Calcular retrasos (Offsets) → Terminal
- Latency calculations → MaxforLive Patch

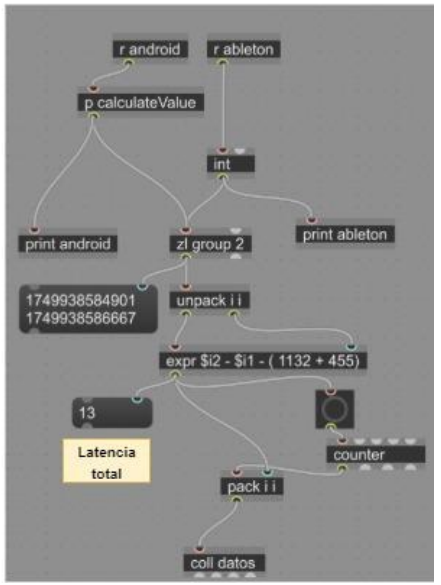
UTC antes del envío (desde Android)

```
fun sendOscTimestamp(serverIP: String, port: Int, oscAddress: String = "latency") {
    Thread {
        try {
            val timestamp = System.currentTimeMillis().toDouble()// en ms
            val message = timestamp.toString()
            Log.d( tag: "HOSTNAME",  msg: "Envio: $message")
```

UTC al recibir (desde el PC)

```
function bang() {
    outlet(0, Date.now());
}
```

Patch de los cálculos



$$\text{Latencia} = (\text{Timestamp}_{PC} - \text{Offset}_{PC}) - (\text{Timestamp}_{Android} - \text{Offset}_{Android})$$

PC

```
C:\Users\lucas>python -c "import ntpl
p.request('pool.ntp.org'); print('Sys
t('Offset     :', r.offset)"
System time: 1749936566.6346269
NTP time    : 1749936566.055647
Offset      : -0.4555182456970215
```

ANDROID

```
~ $ python -c "import ntplib; fr
 = ntplib.NTPClient(); r = ntp.r
print('System time:', time()); p
x_time); print('Offset     :', r
System time: 1749936354.1816916
NTP time    : 1749936355.0312066
Offset      : 1.1322064399719238
```

Result: 47.92 ms average (10 ms and 90 max)

Latency

Tipo de latencia	**Método de medición**	**Valor estimado (ms)**
Transmisión por puerto serie (USB)	3 x 8 bits/ Baudrate (9600)	~2.5
Procesamiento en Arduino	Dificil de medir, obtenido en [26]	—0.2-1.19
Procesamiento en Android	Medido dentro de la aplicación.	1.0 – 2.0
Transmisión por Wi-Fi (UDP)	Calculada con: formula:	~ 47.92 (media), entre 10 y de máximo 96
Latencia total estimada (media)	Suma de las aproximaciones	**~50 – 52**

Result: 50 -52 ms de media

Presupuesto

Cantidad	Componente	Precio unitario	Subtotal
1	Arduino (UNO compatible)	6 €	6 €
1	Móvil (Pixel 3a usado)	(0-100) €	0 €
3	Faders (10k ohms)	1 €	3 €
3	Knobs (10k ohms)	0,60 €	1,80 €
3	Botones tipo pulsador	0,30 €	0,90 €
–	Cables Arduino (jumper)	3 € (pack)	3 €
1	Cable serial port	2 €	2 €
1	Adaptador USB a USB-C	3 €	3 €
		Total estimado	**20,70 €**
		Total estimado + Móvil nuevo	**120,70 €**

Improvements

Latency

- For music applications, latency of less than 20ms is required.

Possible improvements:

- **Optimize message packaging:**
  - Reduce the use of unnecessary padding (null bytes).
  - Debugging packet sizes to make them more efficient.
- Prioritize UDP Packets

Usability

- **Friendlier interface → Redesign of the app to make it more intuitive**
- Quick setup mode → easy to set up, no programmatic configuration required (current)

Proceeds:

- Improve adoption by new users.
- Reduces the learning curve.
- Improve the experience

Improvements

Efficiency

- Add the number of controllers → use multiplexers.
- **Optimize analog inputs → use multiplexers.**
- **Stability and performance**:
  - Improve the application → avoid crashes.
  - Optimize the most efficient → code (memory, CPU).
  - Ensure a smooth experience.

More Tools

- Prototype project → add more parameters and controllers
- Customizable Tools Window
- Add functionalities to connect other devices → not just Arduino (AKAI, Novation, etc)
- Future features:
  - Synchronize via MIDI with other devices
  - Implement musical instruments

Improvements

Two-way communication

- Send messages from your mobile phone to the Arduino
  - Realized but without specific function

Utilities

- Update information to your device
- Implement a configurable screen
- MIDI communication for different musical functions

Conclusions

Proyect:

- Mobile phones → more than tools
  - They are a technological potential
  - **Wasted**
- Active and versatile tool
- **Multitude of sensors, intuitive touch screens, processors**
  → Outperform current computers
- Bringing mobile phones closer to a tool for DIY or electronic projects
- Low-cost and sustainable solution

Personal:

- Develop a communication system between multiple devices and programs:
  - Arduino
  - Android
  - Max for Live
- Active and versatile tool that opens up new opportunities or ideas for recycling and interconnectivity of Arduino with Android phones.
- **Fulfill a dream → Do a final degree project based on recycling**

References

Introduction

- Oct4a-> servidor maquinas 3D app
- UART connection via Android
- Electronic Ears to listen to the forest

Development

- usb-serial -for -android
- midi-wifi- controller
- rotary-buttons

Scenario

- Elego UNO R3
- Serial

Protocols

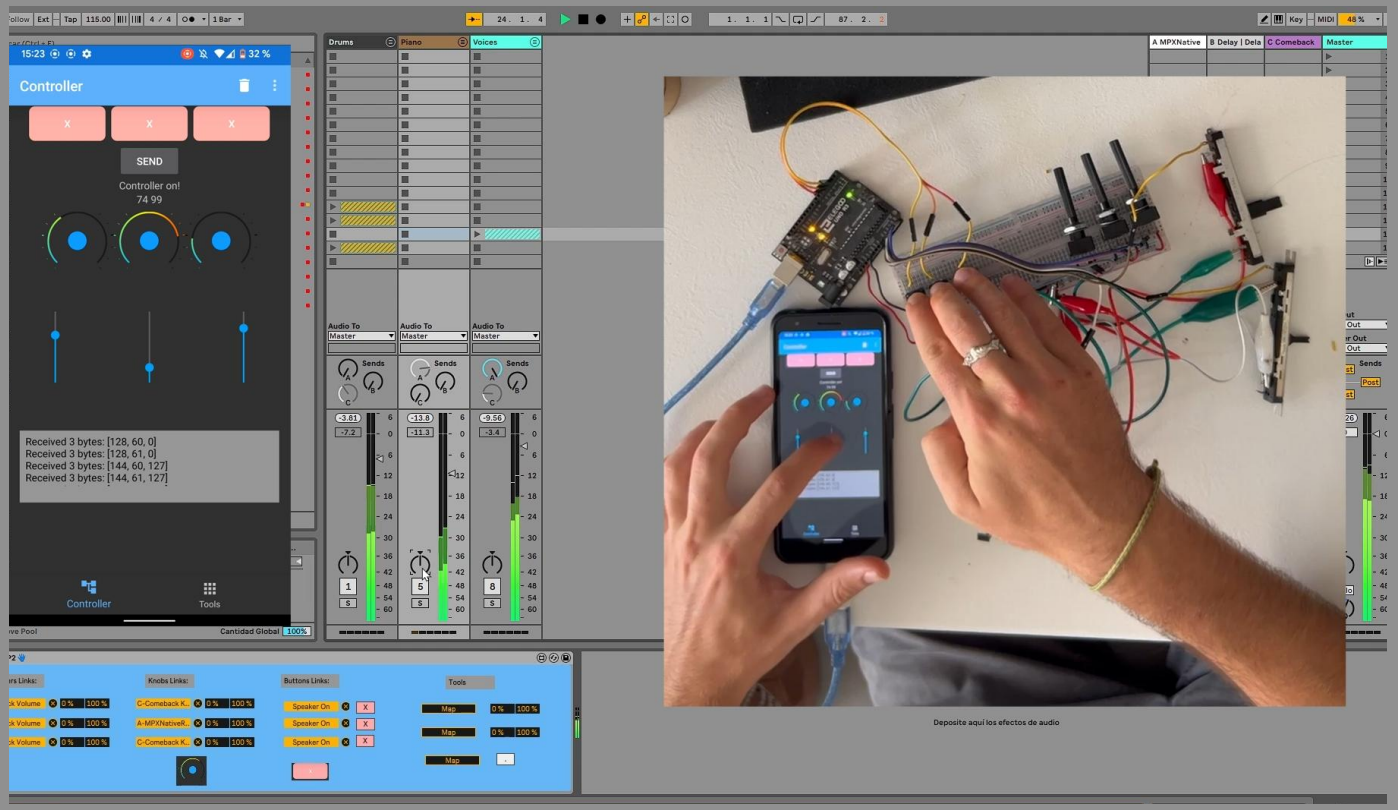
- MIDI
- OSC
- UDP

Images

- Unsplash

Bachelor's Thesis
[13349]

Reuse of a Mobile Phone as a MIDI Controller



Lucas Fuentes-Cantillana Monereo (100429218)

Bachelor's Degree in Sound and Image Engineering

Tutor: Carlos Rodriguez Mayor

Thank you!

Date:
Jul, 8, 2025

Funcionalidades

- **Guardar Rutinas:**
  - Forma de ordenar el tiempo de práctica

- **Ver tu progresión:**
  - Monitorizar de una forma interactiva el progreso para incrementar la productividad y la satisfacción de haber completado lo preestablecido.

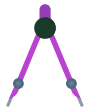
- **Crear rutinas para diferentes instrumentos:**
  - Poder guardar la información de las rutinas de distintos instrumentos

- **Espacio de Herramientas:**
  - Poder usar desde la app diferentes herramientas que implementan funcionalidades de las rutina

  - EJ: practicar escalas a distintas velocidades

- **Espacio para mini-juegos:**
  - Crear juegos interactivos que puedas usar en tus tiempos muertos para mejorar aprender nuevas habilidades o añadir contenido teórica a tus capacidades.

Herramientas



- **Caja de ritmos:**
  - Tener disponible una caja de ritmos simple
  - Posibilidad de cambiar los samples

- **Loops de Bateria:**
  - Tener la samplers de diferentes estilos loopeables (cambiar el tempo)

- **Drones:**
  - Disponer de samples de acordes para practicar encima.

Proyecto

- **Nichos**
  - Dado que dos de nosotros somos músicos, quisimos enfocar nuestro trabajo en un tema relacionado con la elllo.

  - **Tema:** Musical

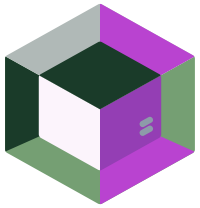
  - **ODS:**
    - **4.** Educación de calidad
    - **9**. Industria, Invocación e Infraestructuras

- **Ideas obtenidas:**
  - **Sampler**
  - **Drum machine**
  - **Piano mobile wifi**
  - **Medir la acústica de la habitación**
  - **App de efectos de voz**

Requerimientos

- **Realizar varias pantallas** → No vale una única pantalla donde se haga todo
  - Drum Machine (Caja de ritmos) → Complejo pero solo una pantalla

- **Requiere distintas aptitudes, no sirve una única y compleja:** →
  El trabajo va sobre aprender las distintas herramientas de una app en Android



- **Conclusión:**
  - Buscar una excusa para realizar todo lo que teníamos pensado →

    - App de instrumentos musicales
    - App de herramientas musicales

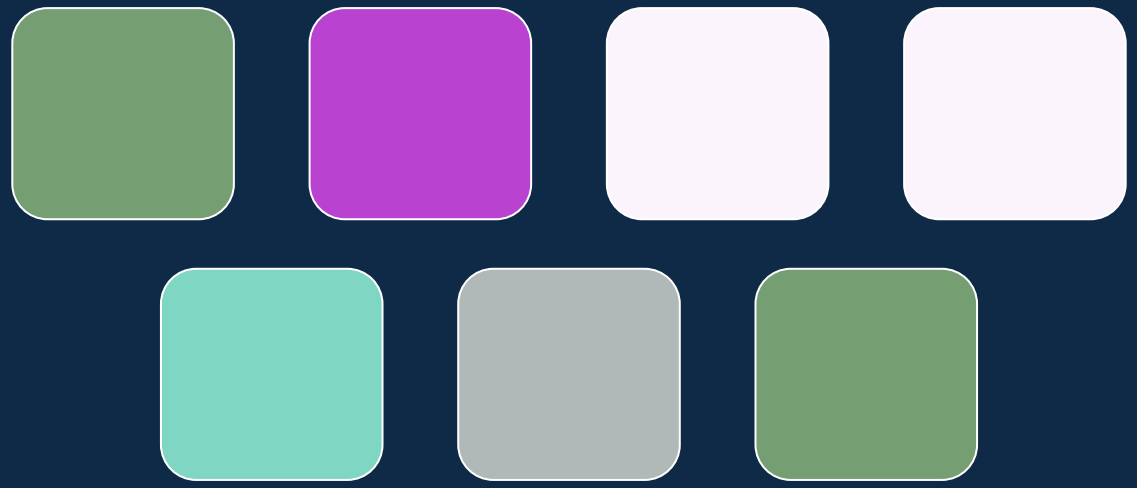
    - **Rutinas Musicales:**
      - Más versatilidad a la hora de las funciones que se pueden adaptar
      - Se amolda muy bien para implementar muchas de la ideas que ya teníamos

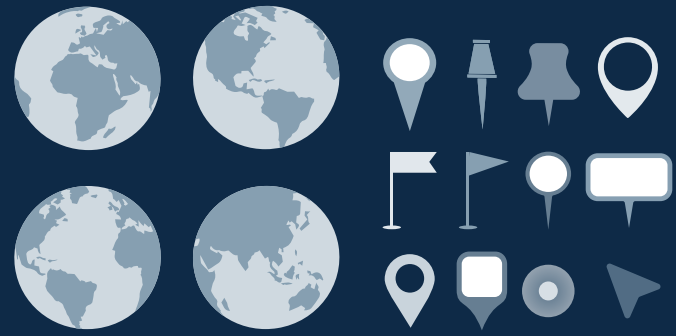
Fonts & colors used

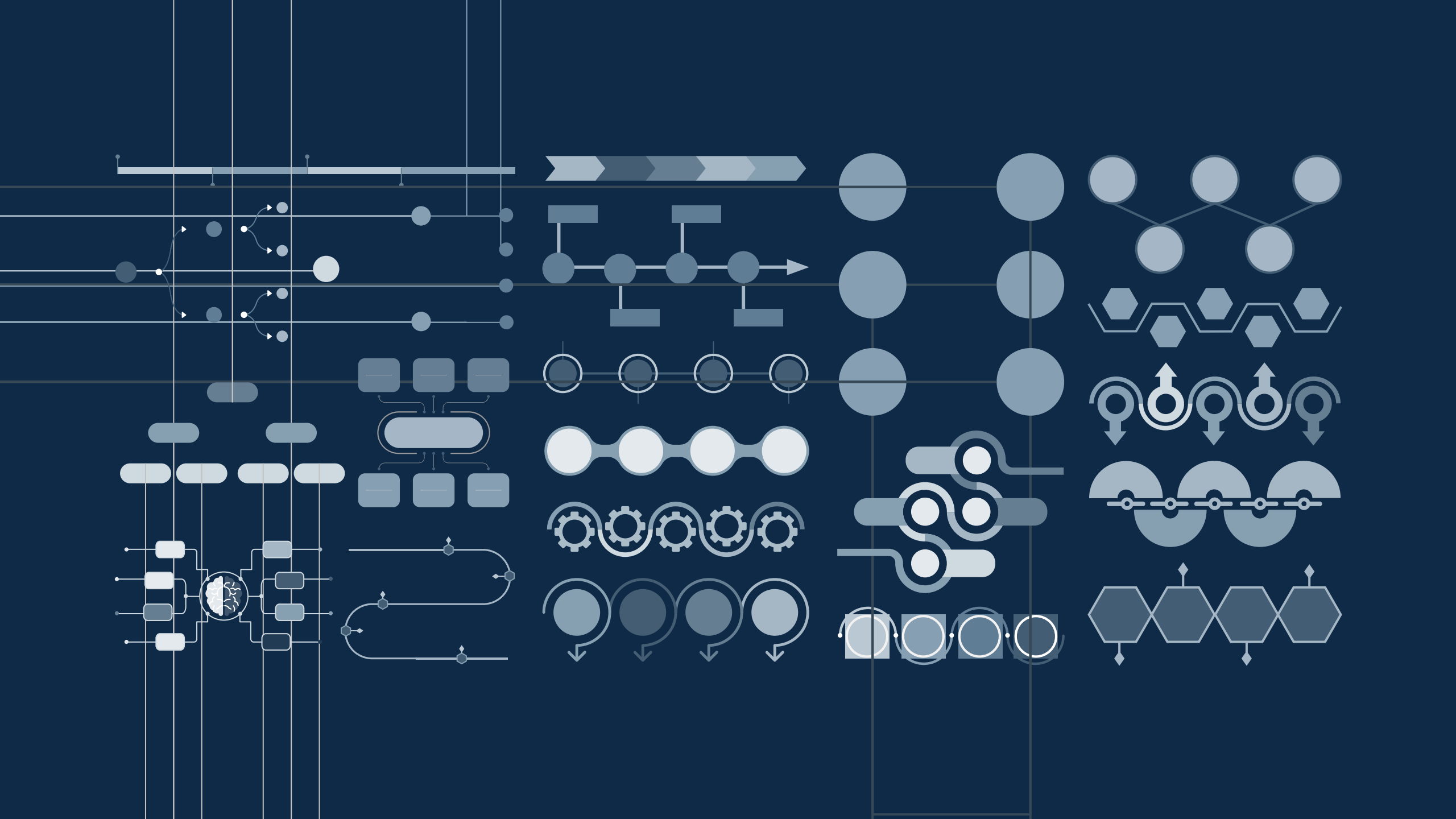
This presentation has been made using the following fonts:

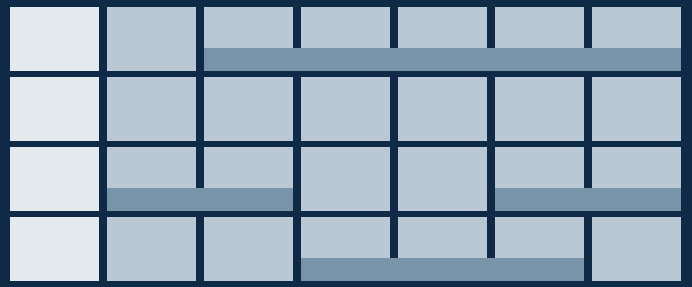
Sora
(https://fonts.google.com/specimen/Sora)

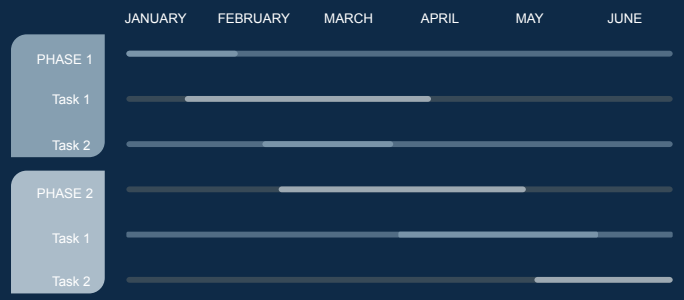
Open Sans
(https://fonts.google.com/specimen/Open+Sans)

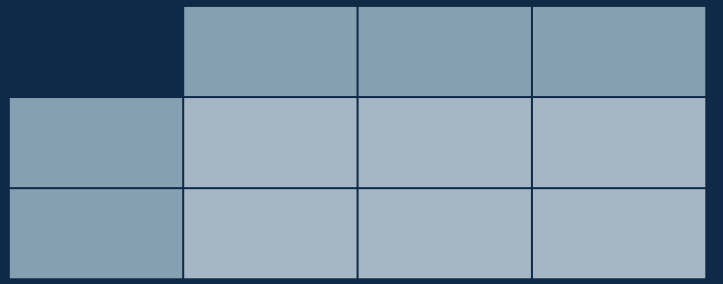


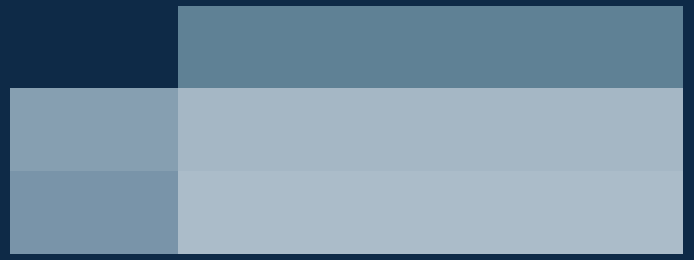


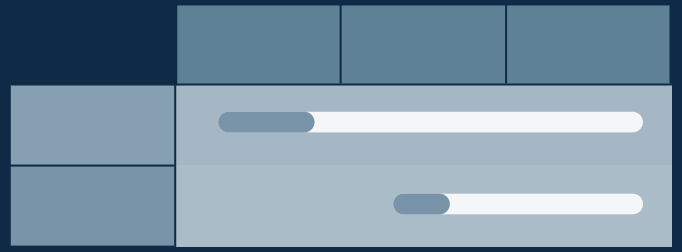


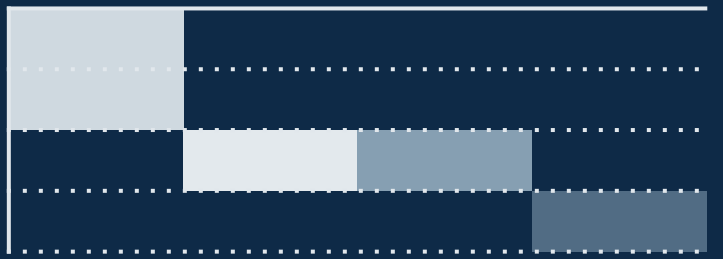


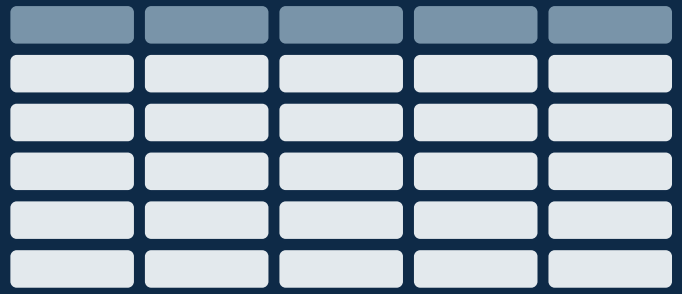

JANUARY
FEBRUARY
MARCH
APRIL
MAY
JUNE
PHASE 1
Task 1
Task 2
PHASE 2
Task 1
Task 2

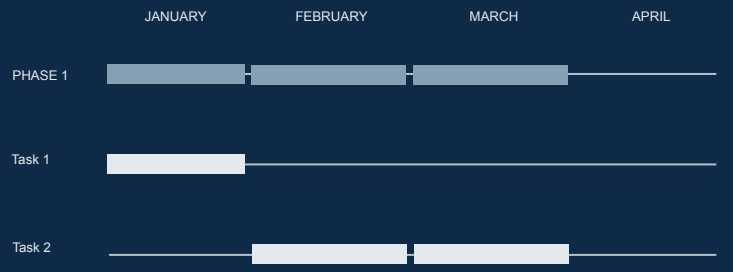


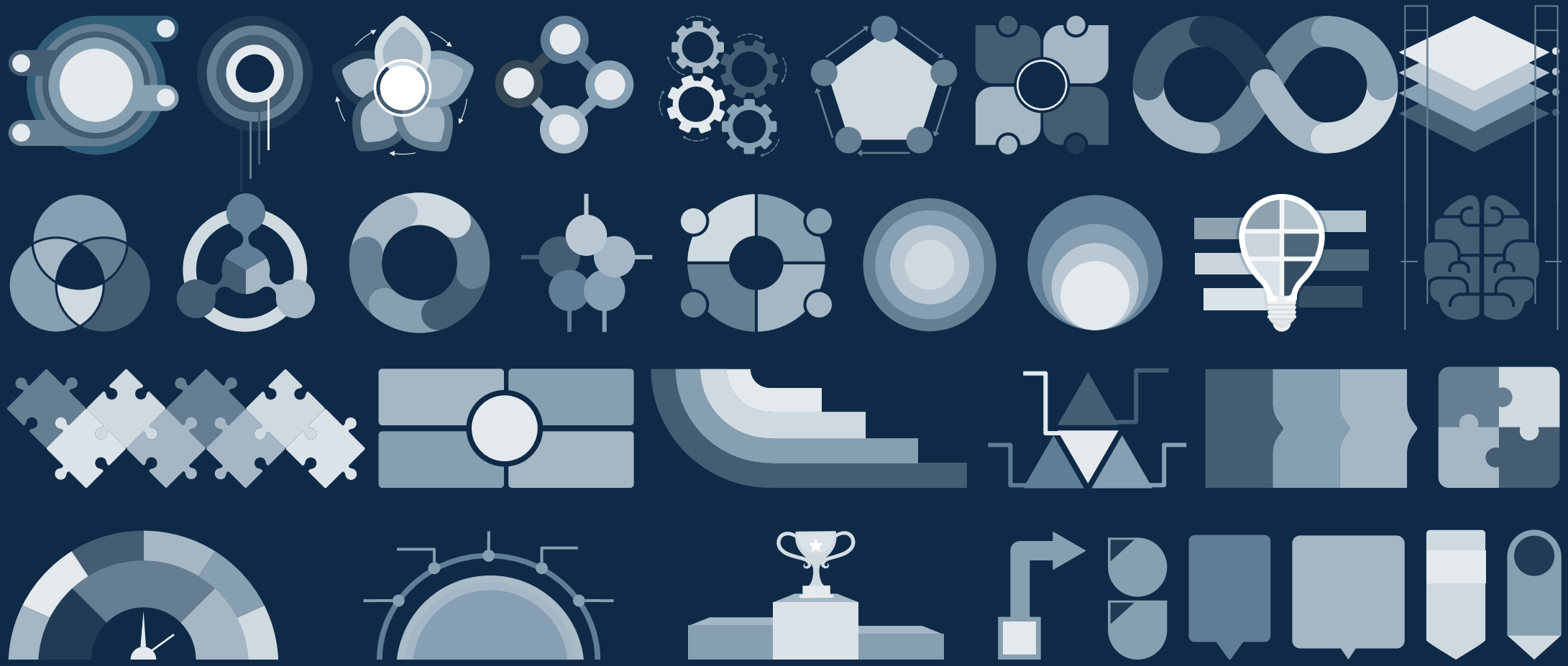









JANUARY
FEBRUARY
MARCH
APRIL
PHASE 1
Task 1
Task 2



...and our sets of editable icons

You can **resize** these icons without losing quality.
You can **change the stroke and fill color**; just select the icon and click on the **paint bucket/pen**.
In Google Slides, you can also use **Flaticon's extension**, allowing you to customize and add even more icons.

Educational Icons

Medical Icons

Business Icons

Teamwork Icons

Help & Support Icons

Avatar Icons

Creative Process Icons

Performing Arts Icons

Nature Icons

SEO & Marketing Icons