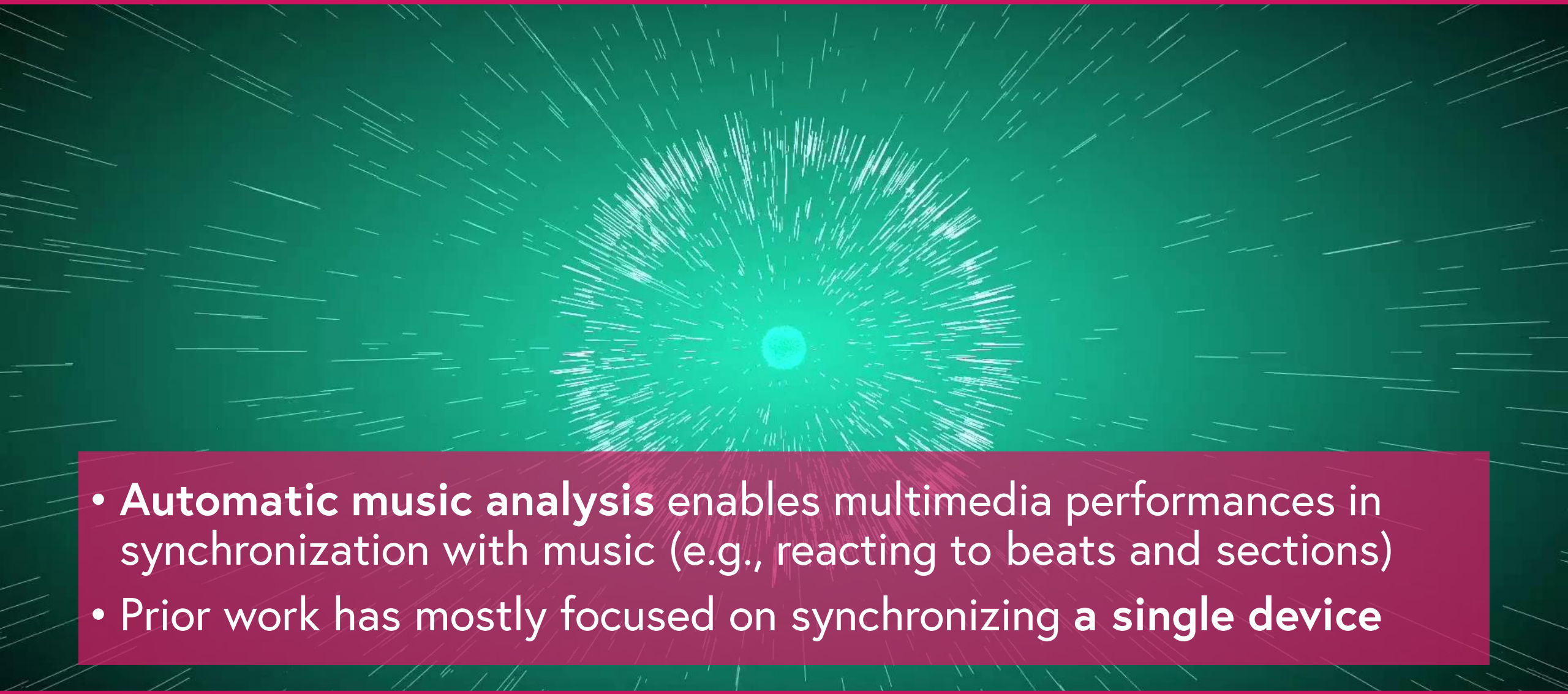
Songle SYNC

A Large-Scale Web-based Platform for Controlling Various Devices in Synchronization with Music

Jun Kato, Masa Ogata, Takahiro Inoue, Masataka Goto

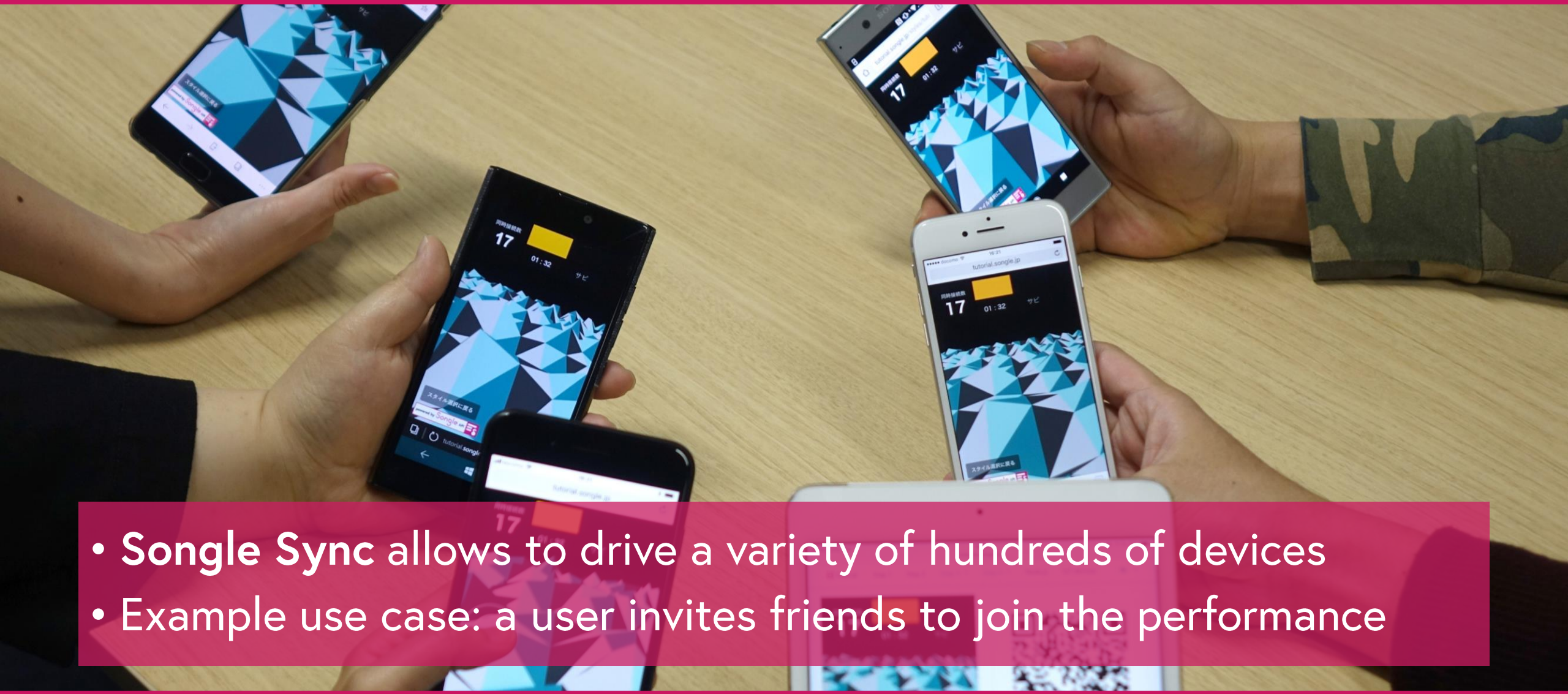
National Institute of Advanced Industrial Science and Technology (AIST)

Music-Driven Multimedia Performance



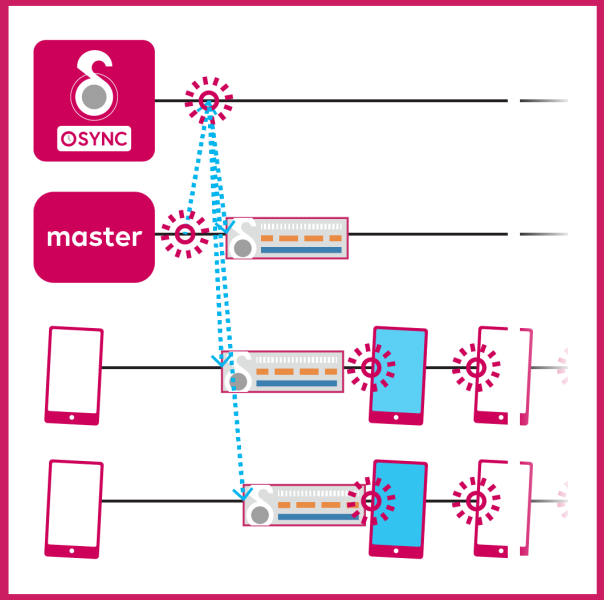
- **Automatic music analysis** enables multimedia performances in synchronization with music (e.g., reacting to beats and sections)
- Prior work has mostly focused on synchronizing **a single device**

Music-Driven Multimedia Performance at Scale

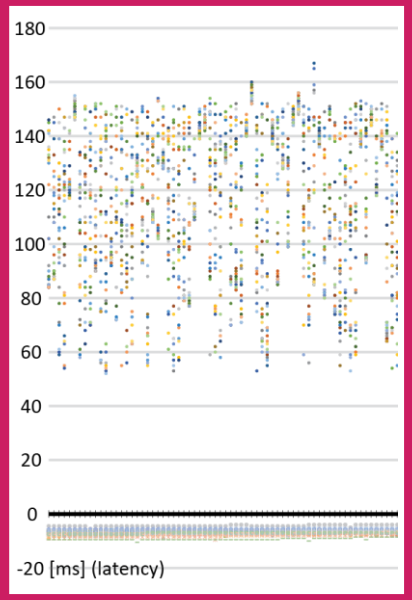


- **Songle Sync** allows to drive a variety of hundreds of devices
- Example use case: a user invites friends to join the performance

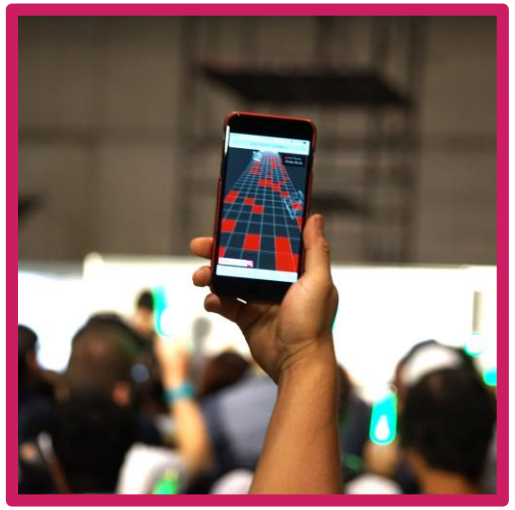
Research contributions (outline of this talk)



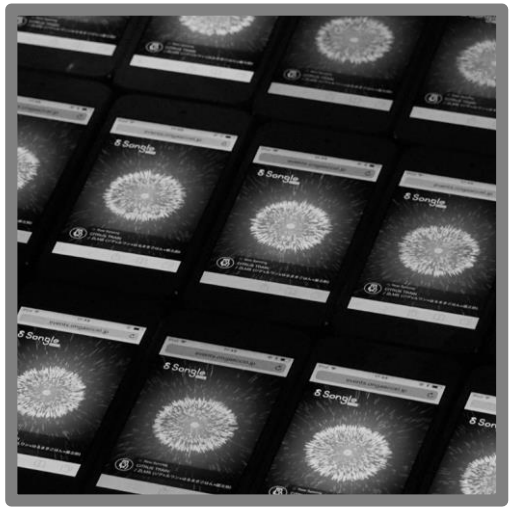

```
import SongleAPI from
"songle-api";

player.on("beatEnter",
listener);
```



Features Architecture Dev. Kit Evaluations



Dynamic
hardware setup

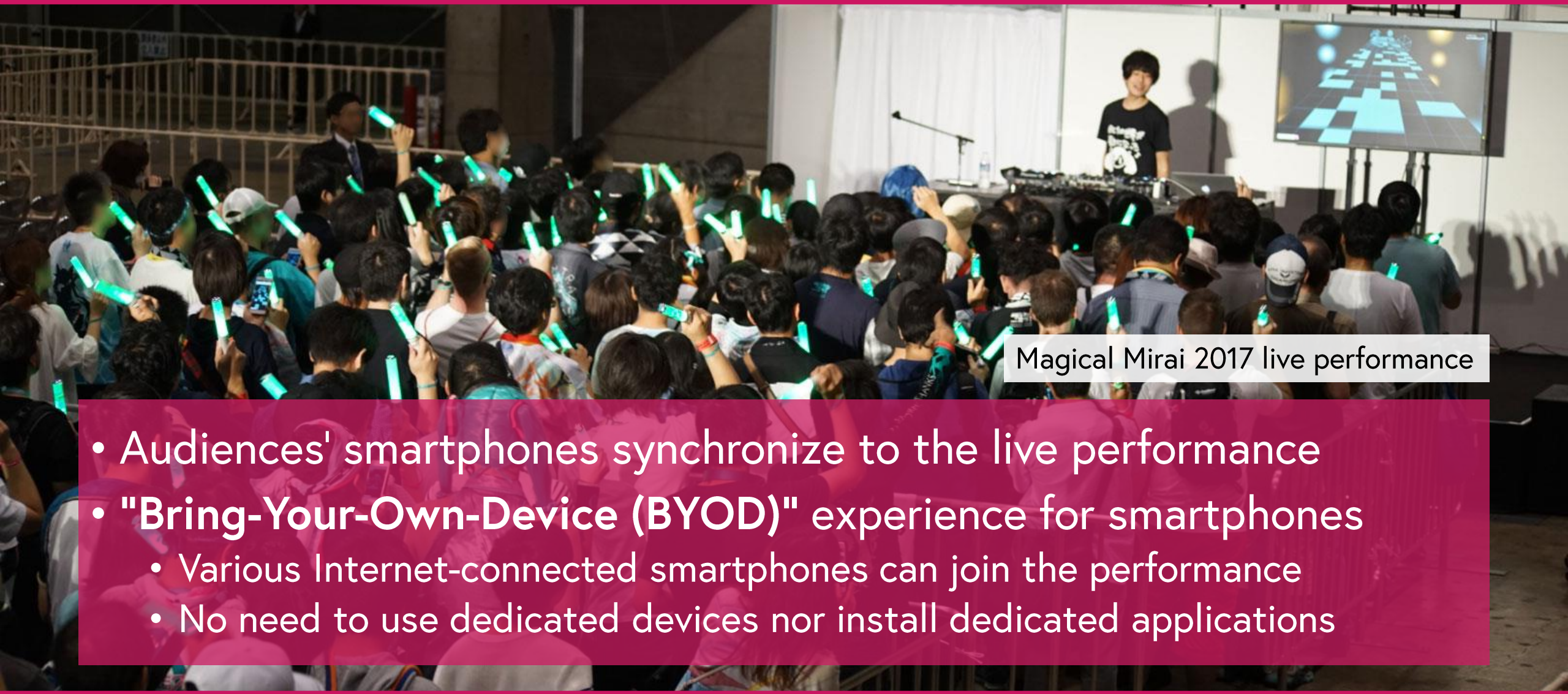


Scalable
device control

Stable
device control

Heterogeneous
hardware setup

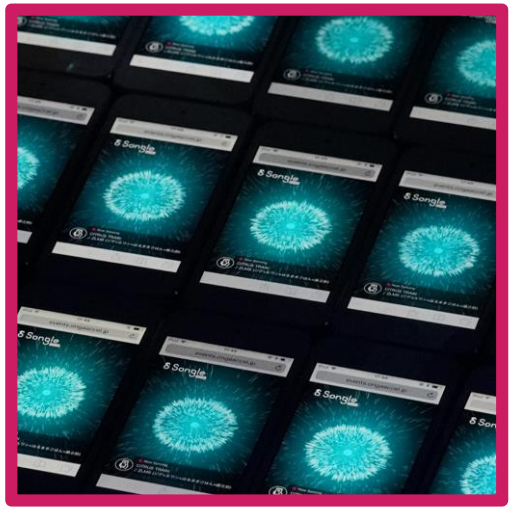
Songle Sync allows dynamic hardware setup



Magical Mirai 2017 live performance

- Audiences' smartphones synchronize to the live performance
- **"Bring-Your-Own-Device (BYOD)"** experience for smartphones
  - Various Internet-connected smartphones can join the performance
  - No need to use dedicated devices nor install dedicated applications

Dynamic
hardware setup



Scalable
device control

Stable
device control

Heterogeneous
hardware setup

Songle Sync provides scalable control of devices

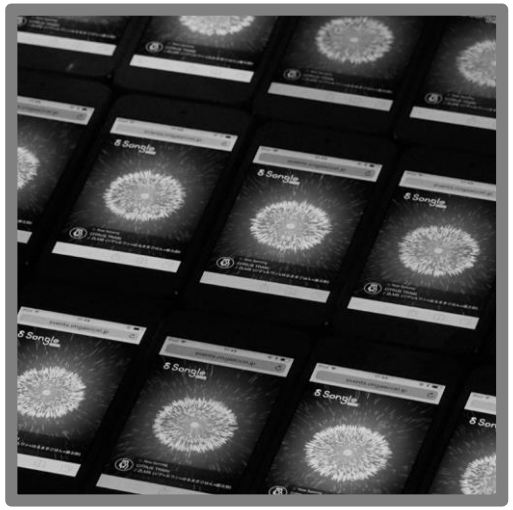
SNOW MIKU LIVE! 2018 pre-event performance

Songle Sync can synchronize hundreds of devices at the same time

Songle Sync provides scalable control of devices

SNOW MIKU LIVE! 2018 pre-event performance

Dynamic
hardware setup



Scalable
device control

Stable
device control

Heterogeneous
hardware setup

Songle Sync provides stable control of devices

Magical Mirai 2018 performance augmenting the whole event venue

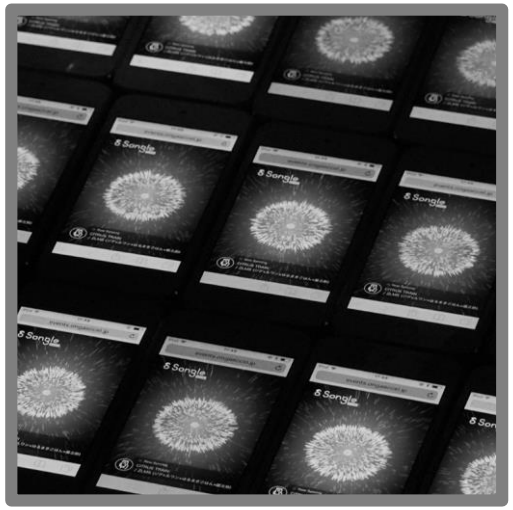
Songle Sync can synchronize devices under challenging networking environments (e.g., smartphones with slow wireless network)

Songle Sync provides stable control of devices

Magical Mirai 2018 performance augmenting the whole event venue

Music-Driven Multimedia Performance at Scale

Dynamic
hardware setup

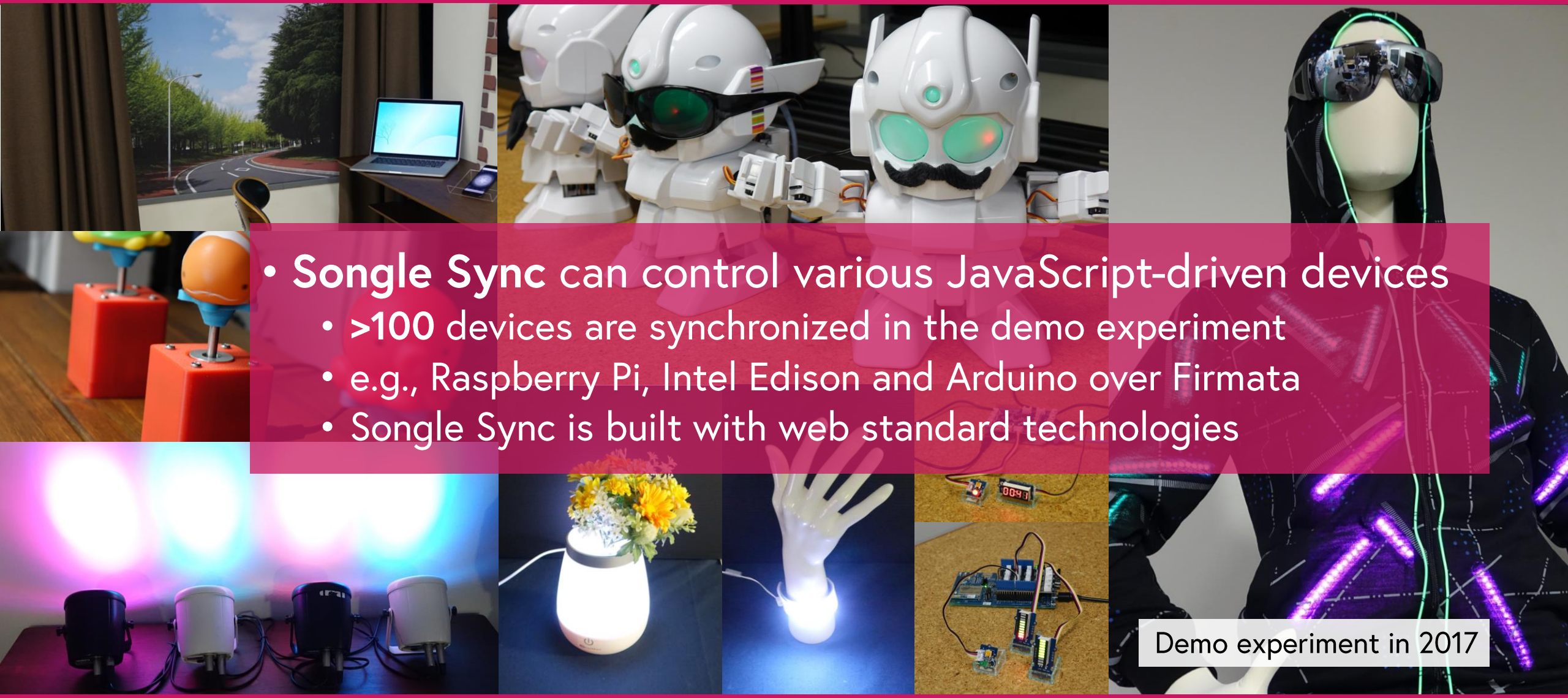


Scalable
device control

Stable
device control

Heterogeneous
hardware setup

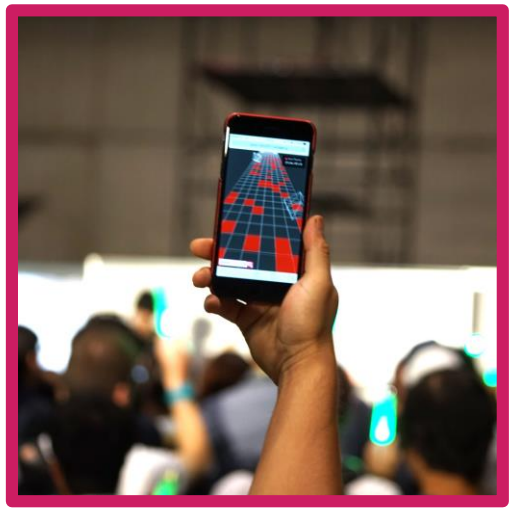
Songle Sync allows heterogeneous hardware setup



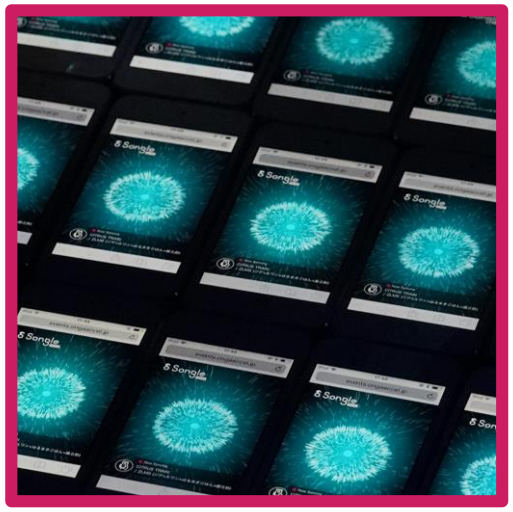
- **Songle Sync** can control various JavaScript-driven devices
  - **>100** devices are synchronized in the demo experiment
  - e.g., Raspberry Pi, Intel Edison and Arduino over Firmata
  - Songle Sync is built with web standard technologies

Demo experiment in 2017

Music-Driven Multimedia Performance at Scale



Dynamic
hardware setup



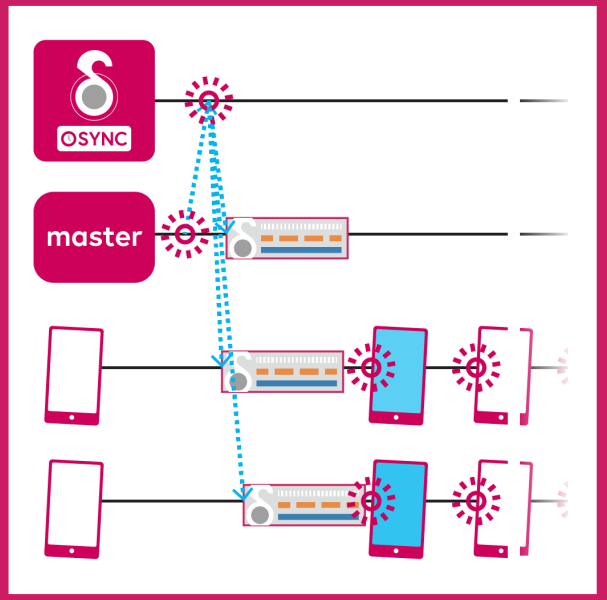
Scalable
device control

Stable
device control

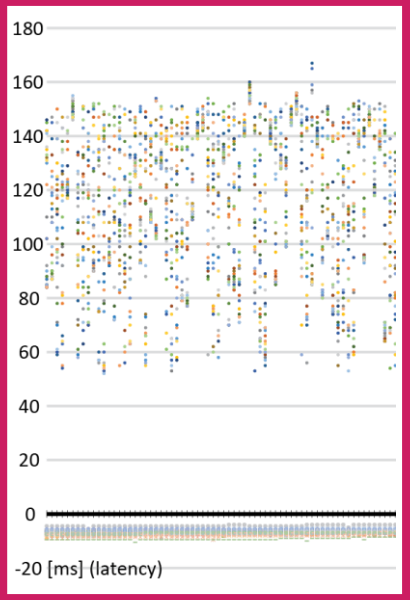
Heterogeneous
hardware setup

Q. How did we enable these features?

Research contributions (outline of this talk)

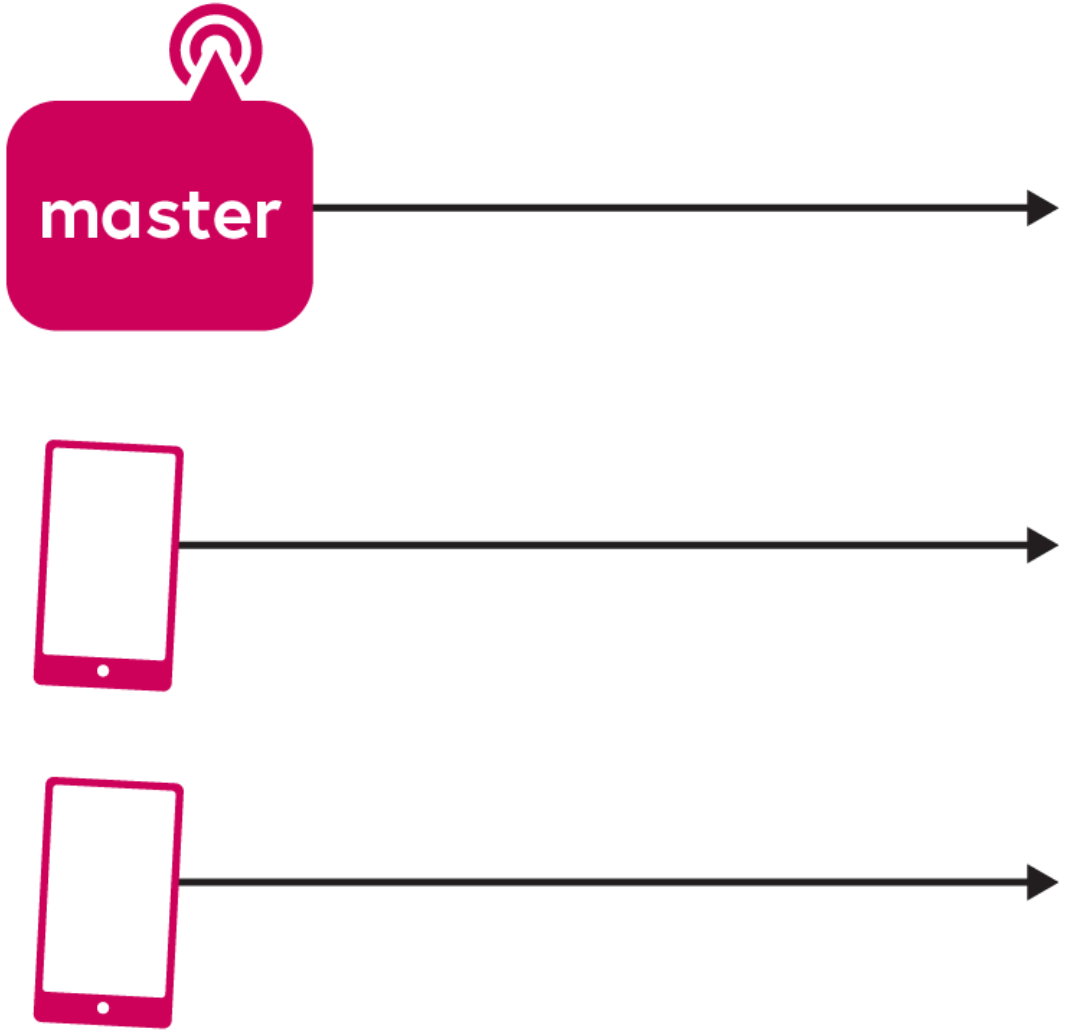



```
import SongleAPI from
"songle-api";

player.on("beatEnter",
listener);
```



Features **Architecture** Dev. Kit Evaluations

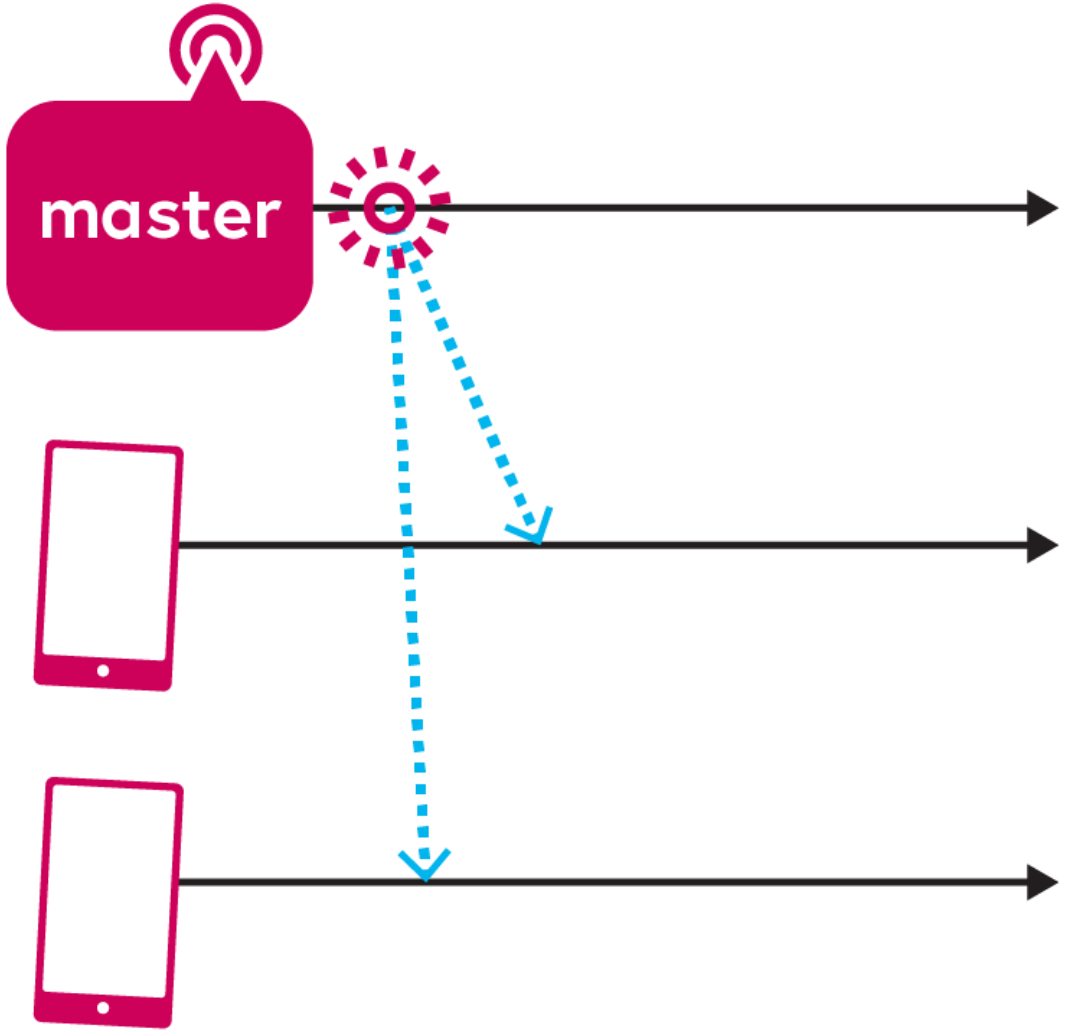
Conventional "Always-On" Architecture



- Suppose you want to flash smartphone screens at each beat of a musical piece…

- Horizontal axis = time
- **Master**: a node that knows timings
- **Slaves (smartphones)**: nodes that are expected to flash their screens synchronously

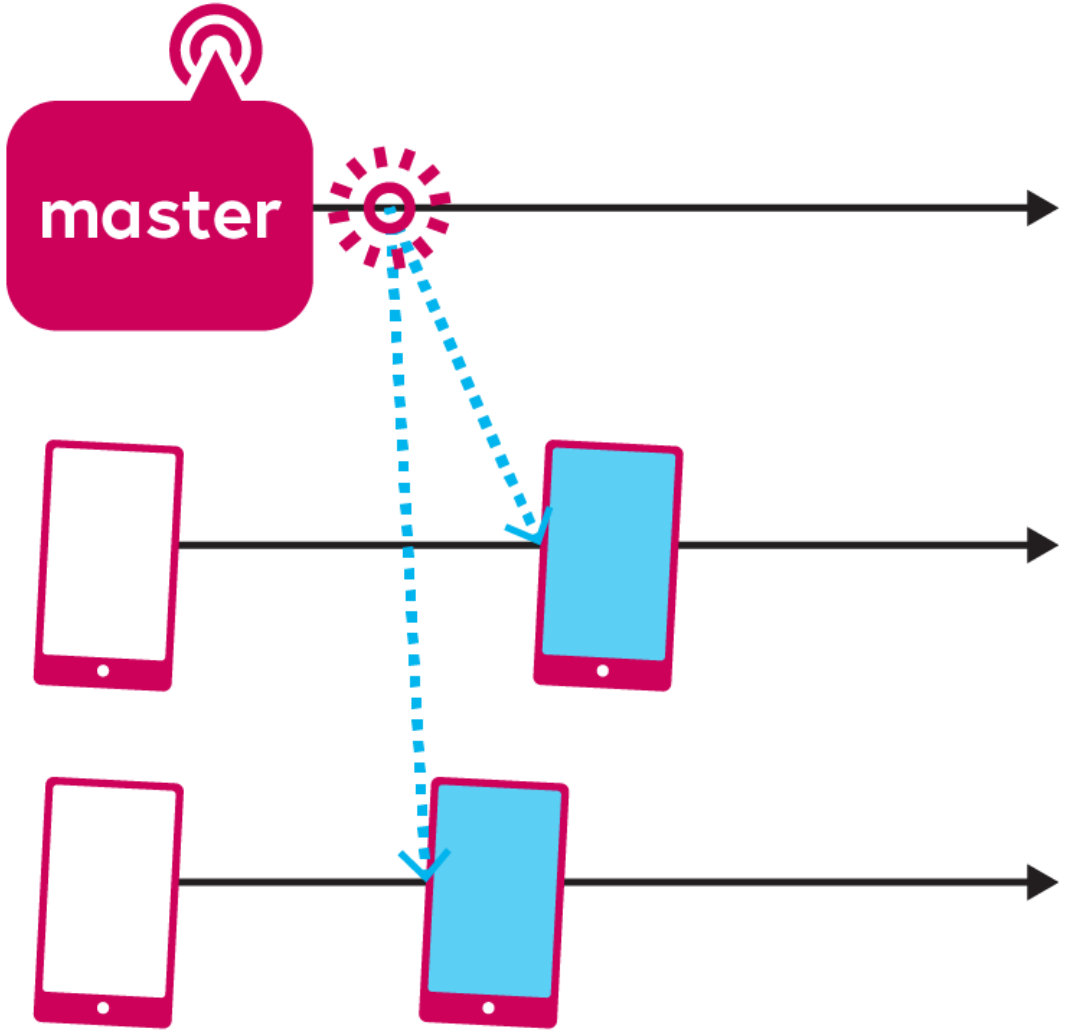
Conventional "Always-On" Architecture



- Suppose you want to flash smartphone screens at each beat of a musical piece…

1. The master node **emits a command at each beat**

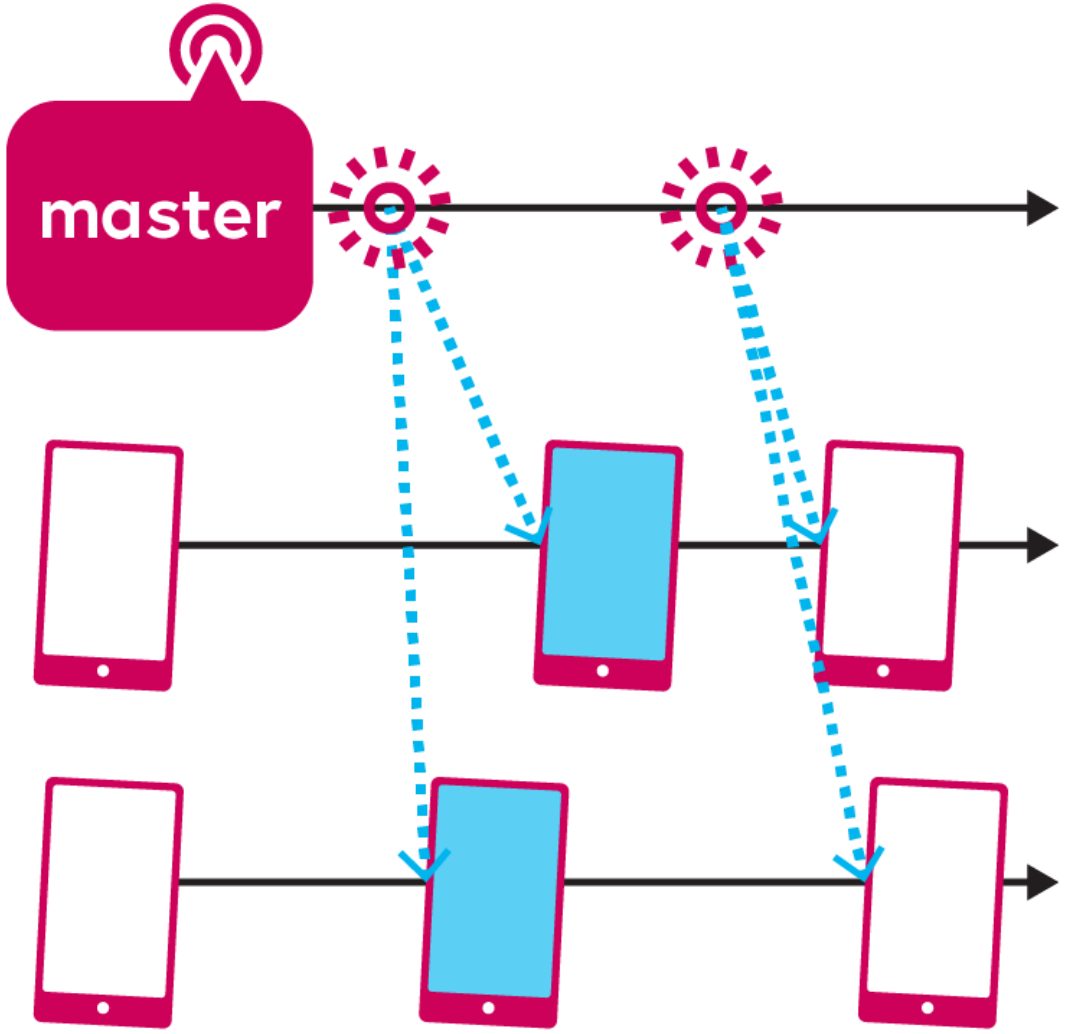
Conventional "Always-On" Architecture



- Suppose you want to flash smartphone screens at each beat of a musical piece…

1. The master node **emits a command at each beat**
2. Each slave node **reacts to the command** by a screen flash

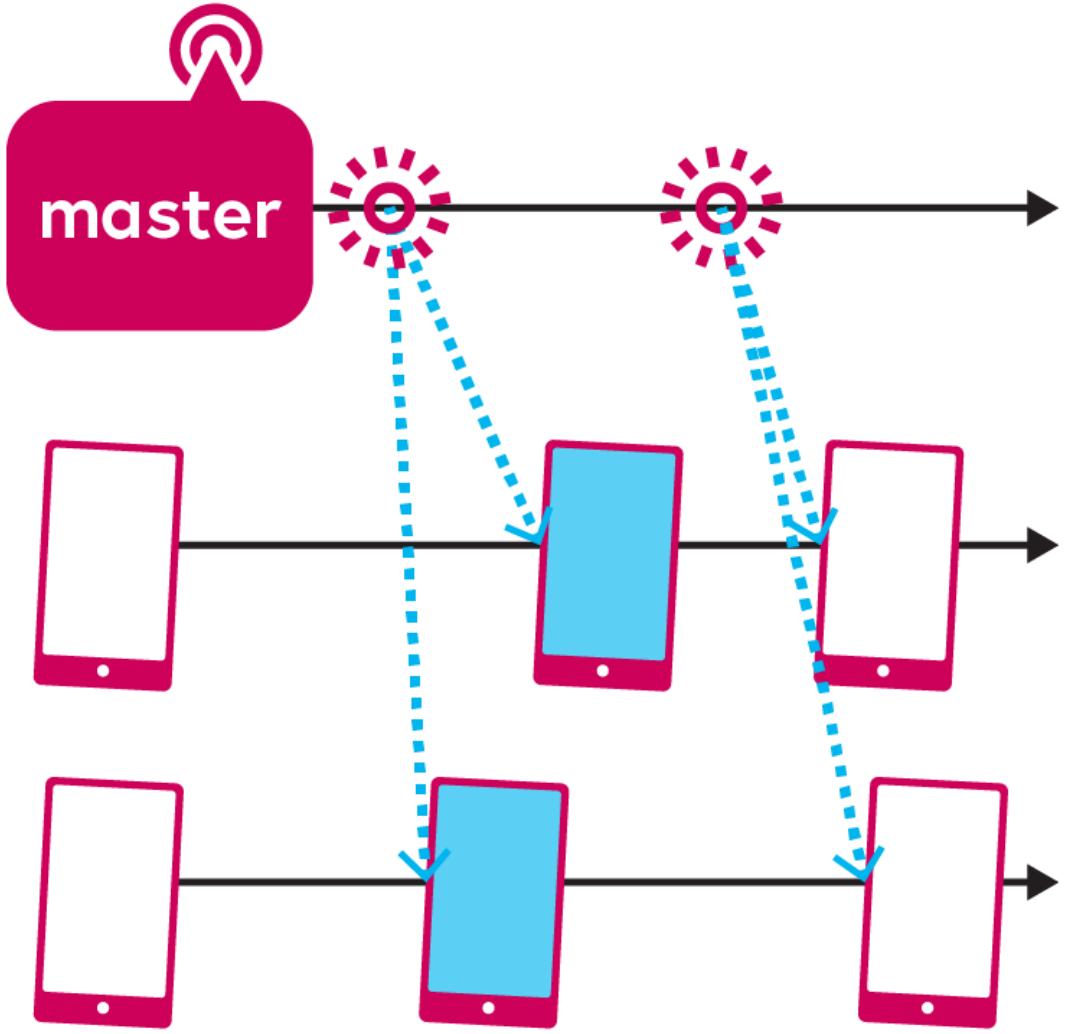
Conventional "Always-On" Architecture



- Suppose you want to flash smartphone screens at each beat of a musical piece…

1. The master node **emits a command at each beat**
2. Each slave node **reacts to the command** by a screen flash
3. Repeat 1-2. This **"always-on"** architecture has been used in conventional performances

Conventional "Always-On" Architecture

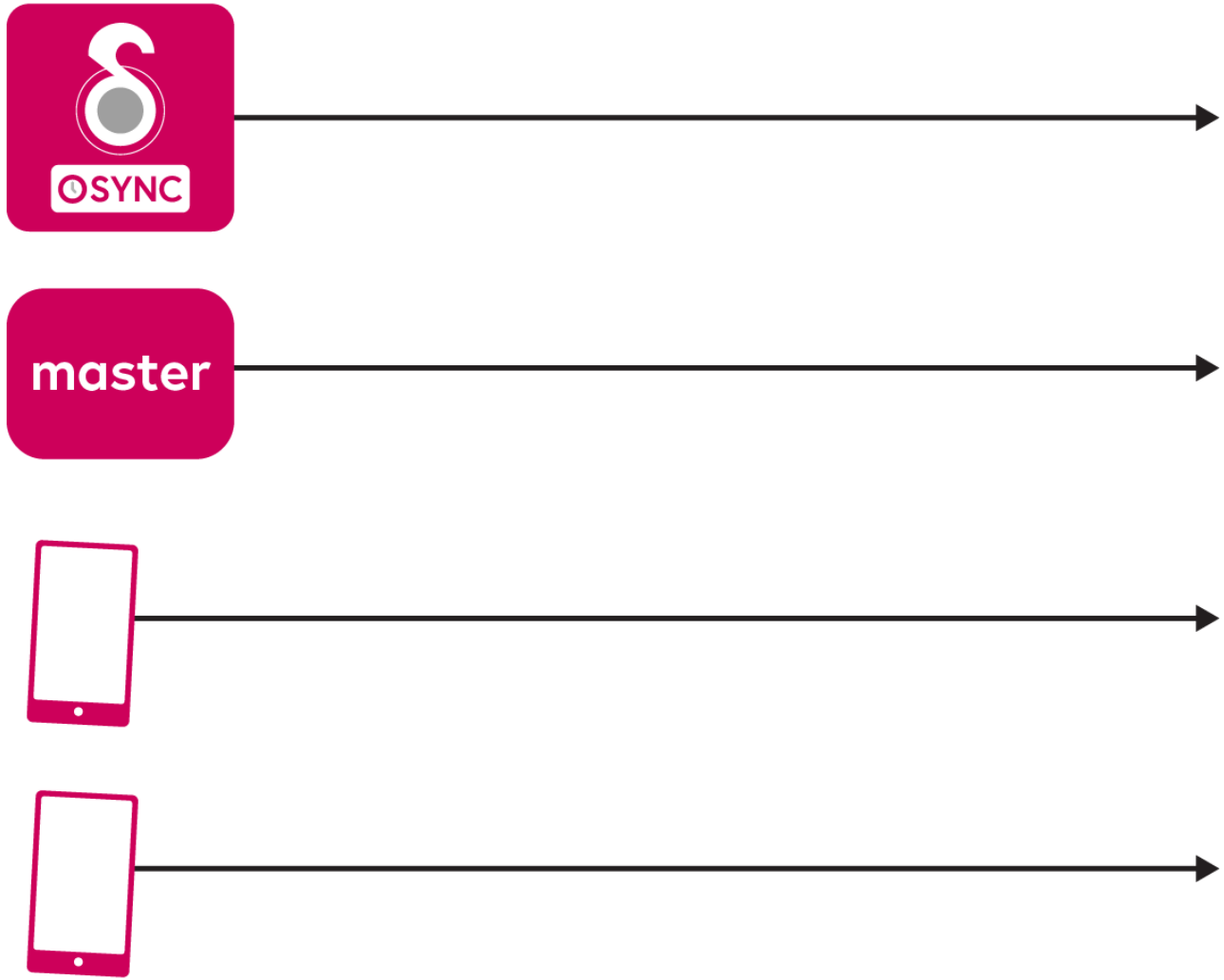


- Suppose you want to flash smartphone screens at each beat of a musical piece…

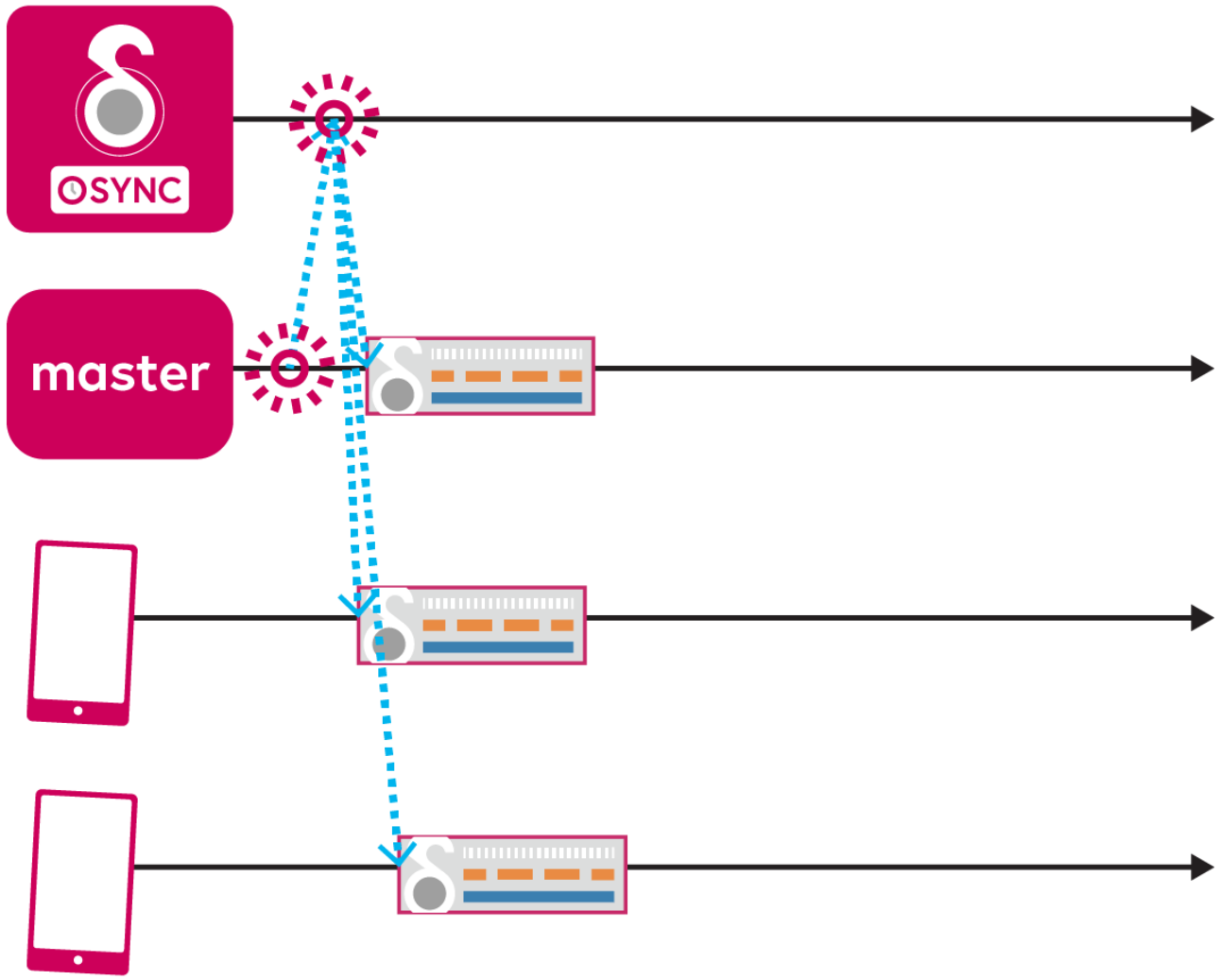
Per-event communication

Inevitable **latency and jitter**

Our Novel "Autonomous Control" Architecture

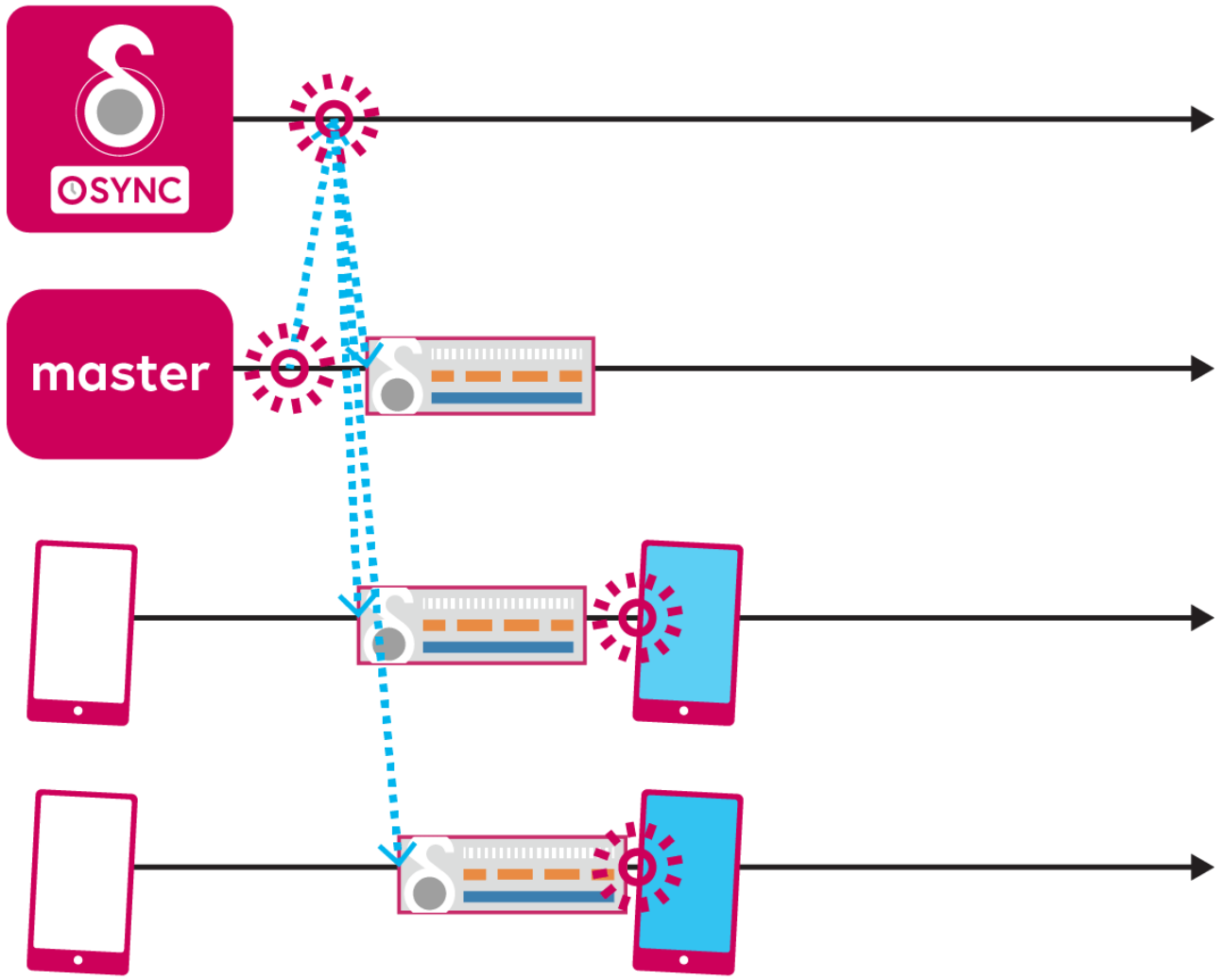


Our Novel "Autonomous Control" Architecture



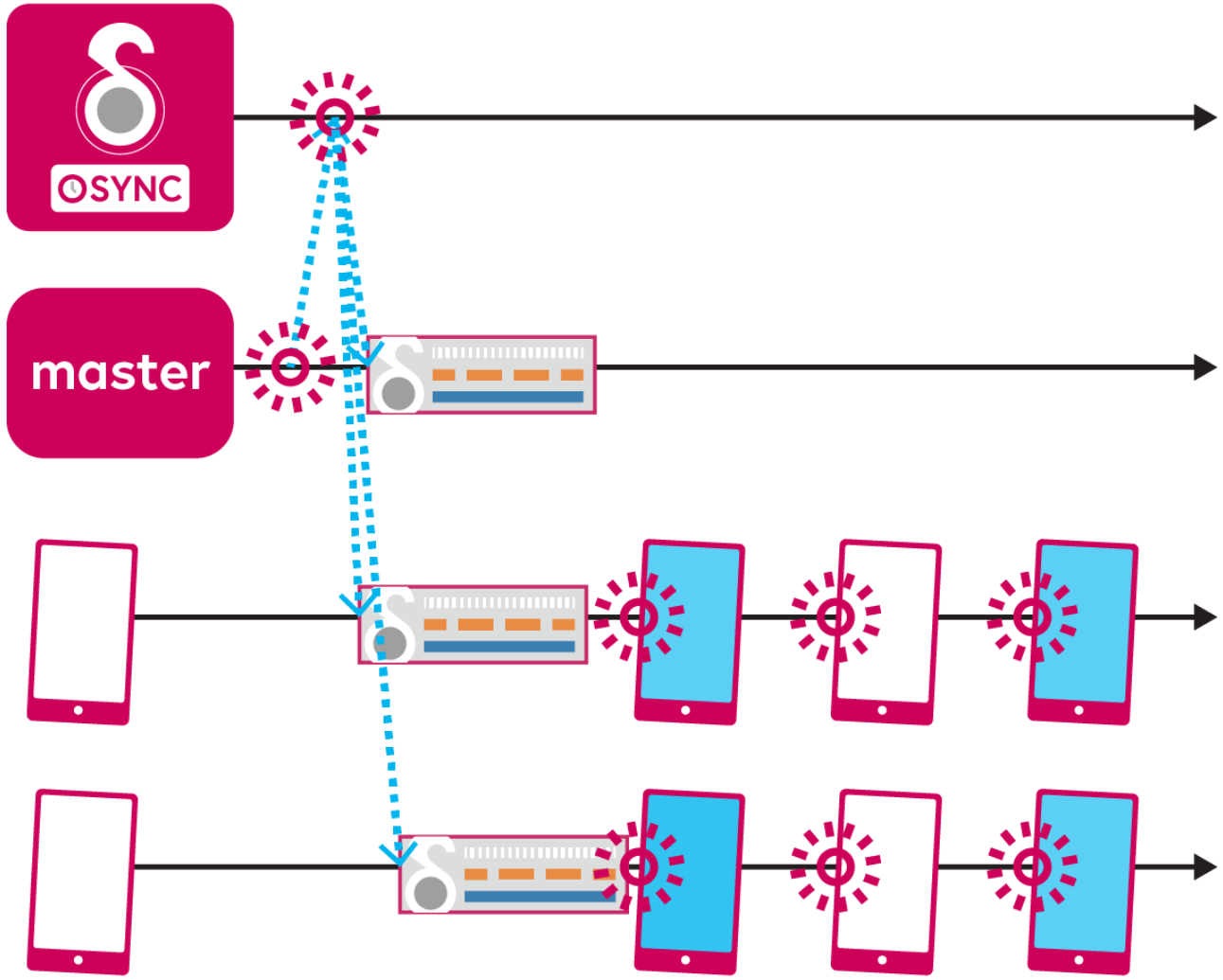
1. The master node chooses a musical piece
2. Songle Sync **distributes timings of beat events**

Our Novel "Autonomous Control" Architecture



1. The master node chooses a musical piece
2. Songle Sync **distributes timings of beat events**
3. Each node knows when to flash the screen

Our Novel "Autonomous Control" Architecture

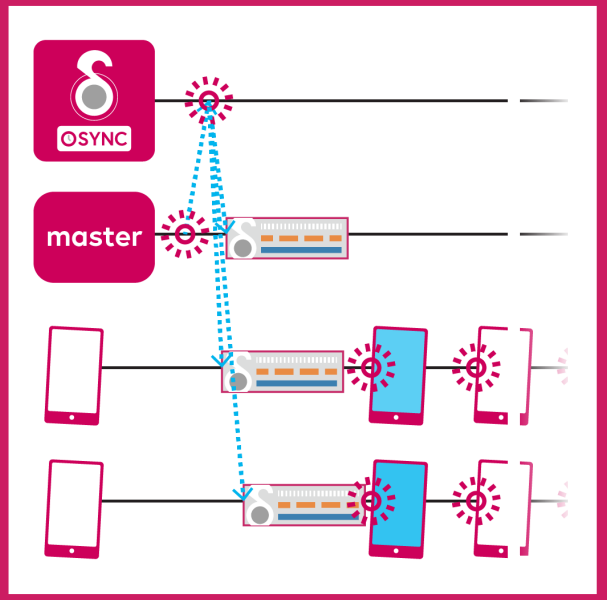


No need for per-event communication

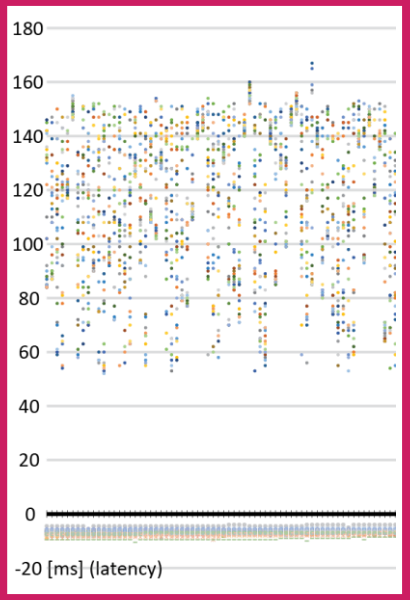
+ **NTP-like protocol** to synchronize clocks

Theoretically **no latency and jitter**

Research contributions (outline of this talk)

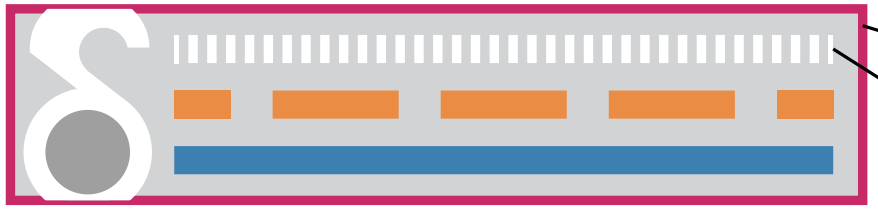



```
import SongleAPI from
"songle-api";

player.on("beatEnter",
listener);
```



Features Architecture **Dev. Kit** Evaluations

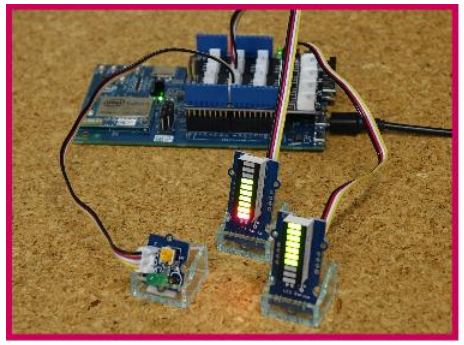
Development Kit (Open Platform)

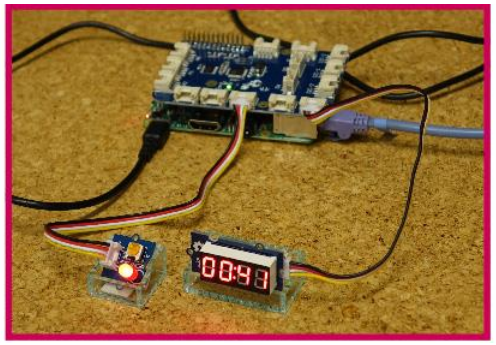


```
player.on("play", listener);
player.on("beatEnter", listener);
                                ...
```

Event-driven APIs for easily synchronizing applications to music playback

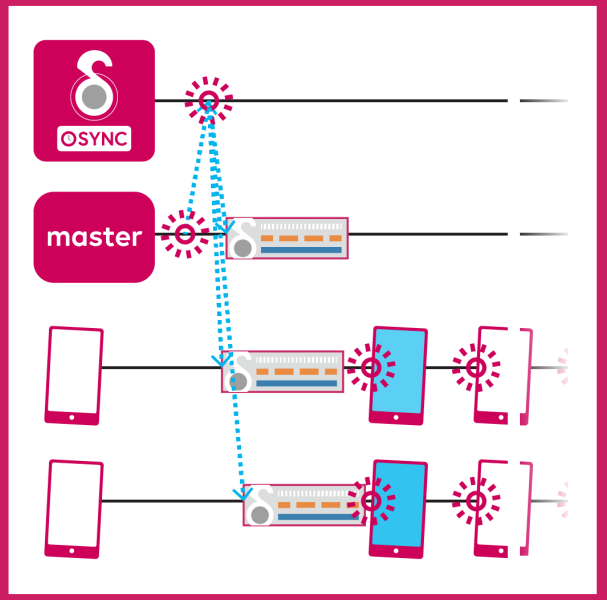
- The code written for one device can drive hundreds of devices synchronously
- No need to worry about networking and synchronization



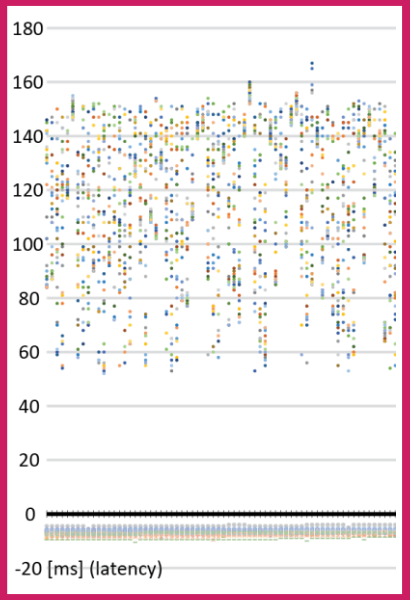


Example programs and **interactive tutorials** to kickstart the development

Research contributions (outline of this talk)




```
import SongleAPI from
"songle-api";

player.on("beatEnter",
listener);
```



Features Architecture Dev. Kit **Evaluations**

1 Performance Evaluations

Network traffic measurement:

500 kB initial load + **30 kB** timings information + **5 kB/min**
Clock/Event Sync **>>** typical video streaming **7.5MB/min**

1 master and **30** slave nodes on VMs with **100** ± **30**[ms] jitter and latency

-20 ~ 0 [ms] (ours) **vs** **40 ~ 180** [ms] latency

7 smartphones/tablets with **Wi-Fi/4G-LTE** connections

web browser screens captured by a 960-fps camera

observed jitters **< 100** [ms] regardless of connection types

2 Deployments in the Wild

- A demo experiment > 110 heterogeneous hardware devices
- A live performance with at least 275 synchronized smartphones

3 Development Kit Usability

- 2-day hackathon with 24 univ. students
- All 6 groups prototyped working apps

Evaluations

1 Performance Evaluations

Songle Sync outperformed "always-on" architecture in both emulated and actual environments

2 Deployments in the Wild

Songle Sync could synchronize a variety of hundreds of devices

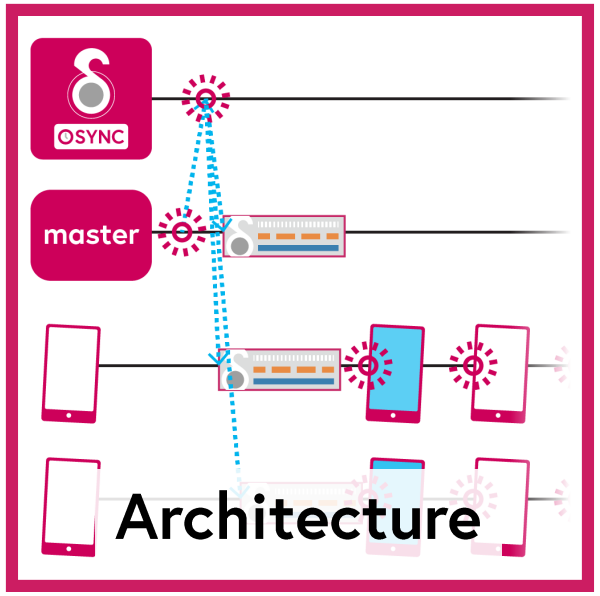
(latest result in the next slide!)

3 Development Kit Usability

Development kit was informative enough

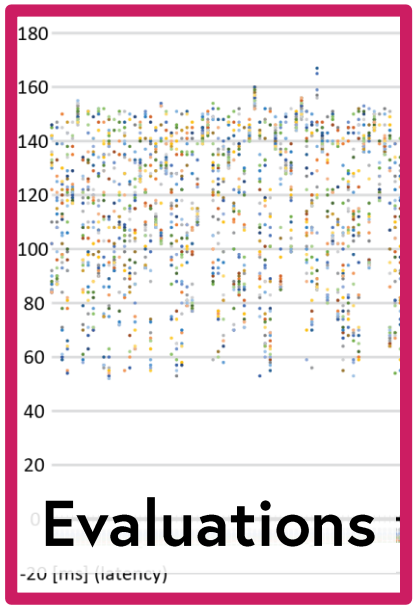
In our recent experiment…

Four research contributions of Songle Sync and its future




```
import SongleAPI from
"songle-api";

player.on("beatEnter",
listener);
```

Development Kit



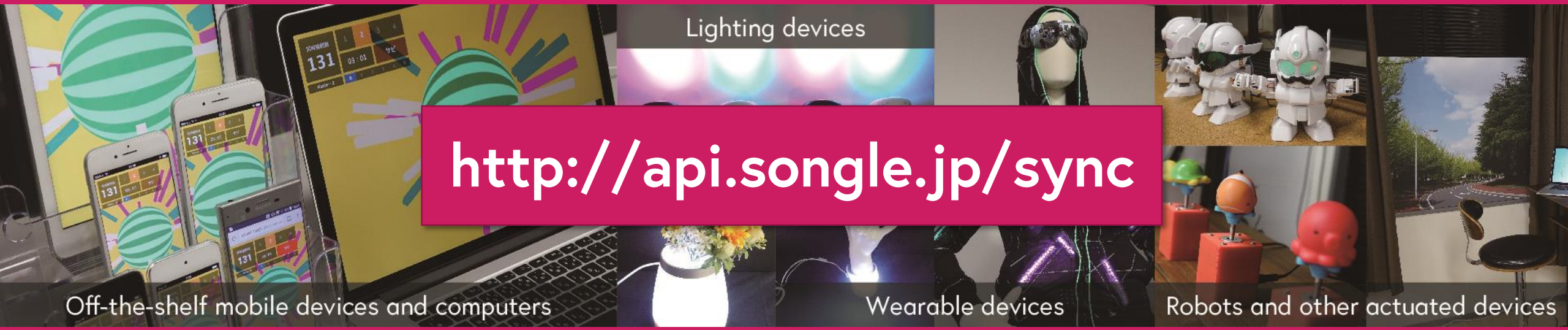
Songle Sync powers the era of **"Internet of Musical Things (IoMT)"**

Start building IoMT applications with Songle Sync!

➡ http://api.songle.jp/sync

Songle SYNC

A Large-Scale Web-based Platform for Controlling Various Devices in Synchronization with Music



Jun Kato, Masa Ogata, Takahiro Inoue, Masataka Goto

National Institute of Advanced Industrial Science and Technology (AIST)