Digitizing a Pinball machine

Foreword

- Some ideas taken from BLEnky
- Focus on general principle

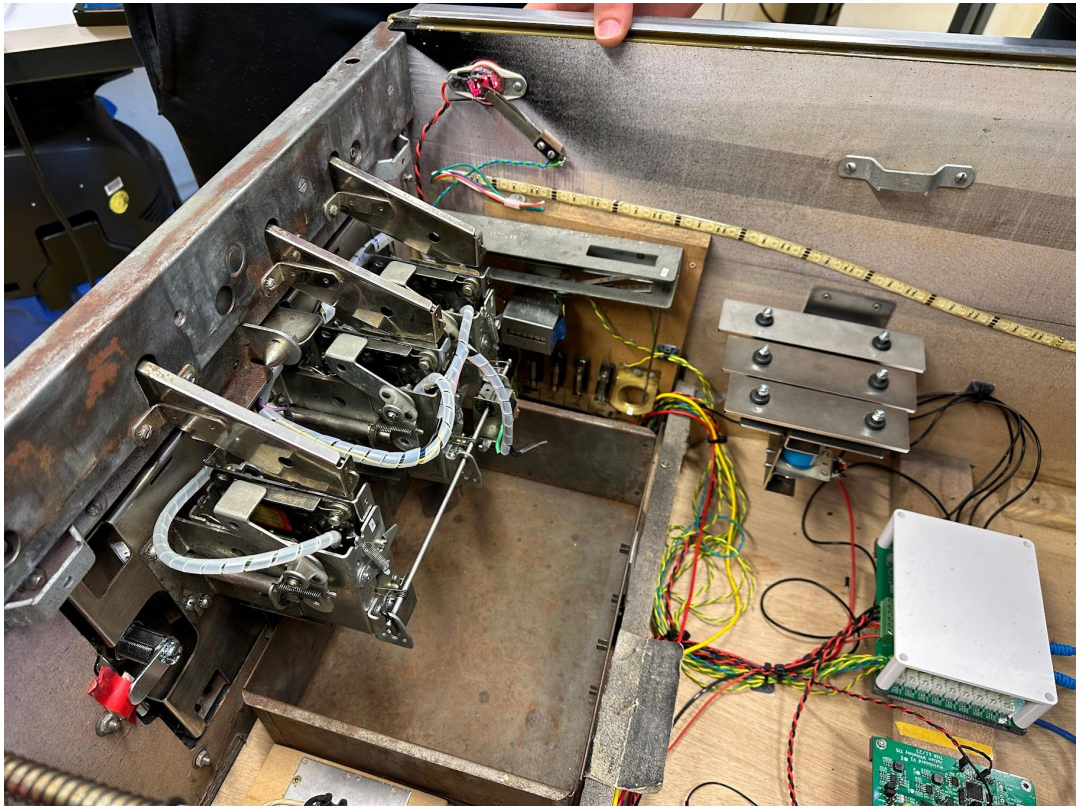
Machine in question

Sensors

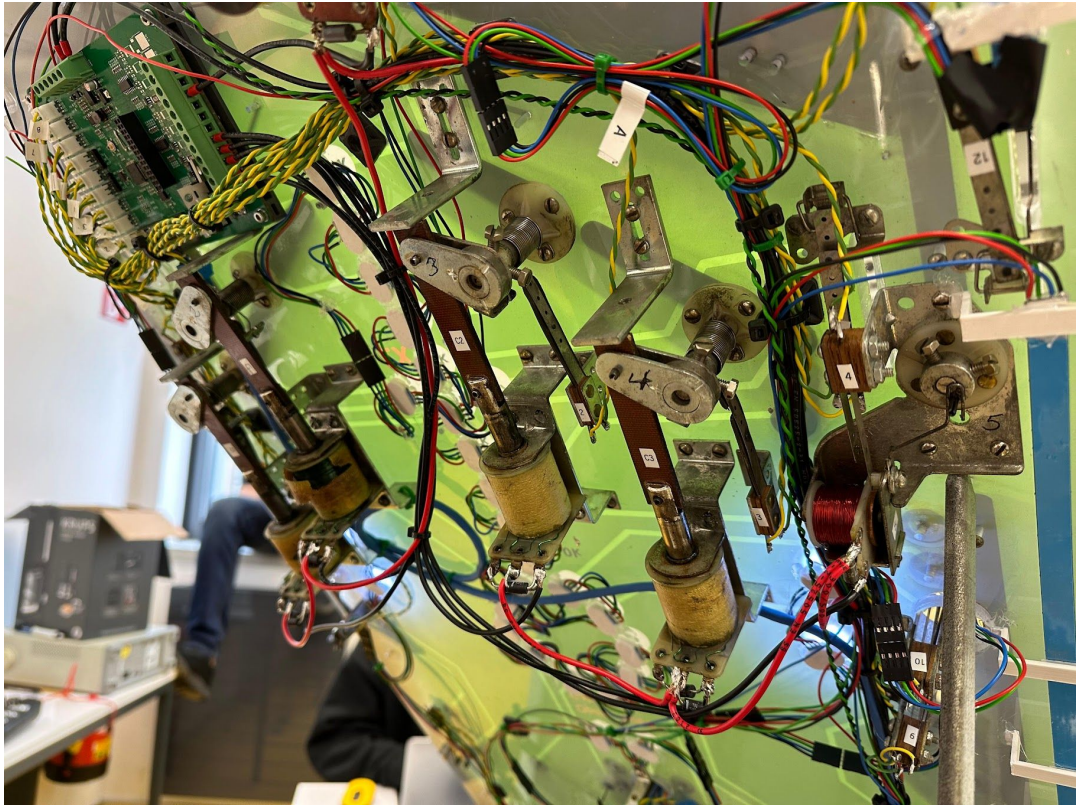

10
100
OPENS
GATE
when lit
500 POIN


10
POINTS
WHEN LIT
1000 POINTS
ADVANCE BO

Sensors

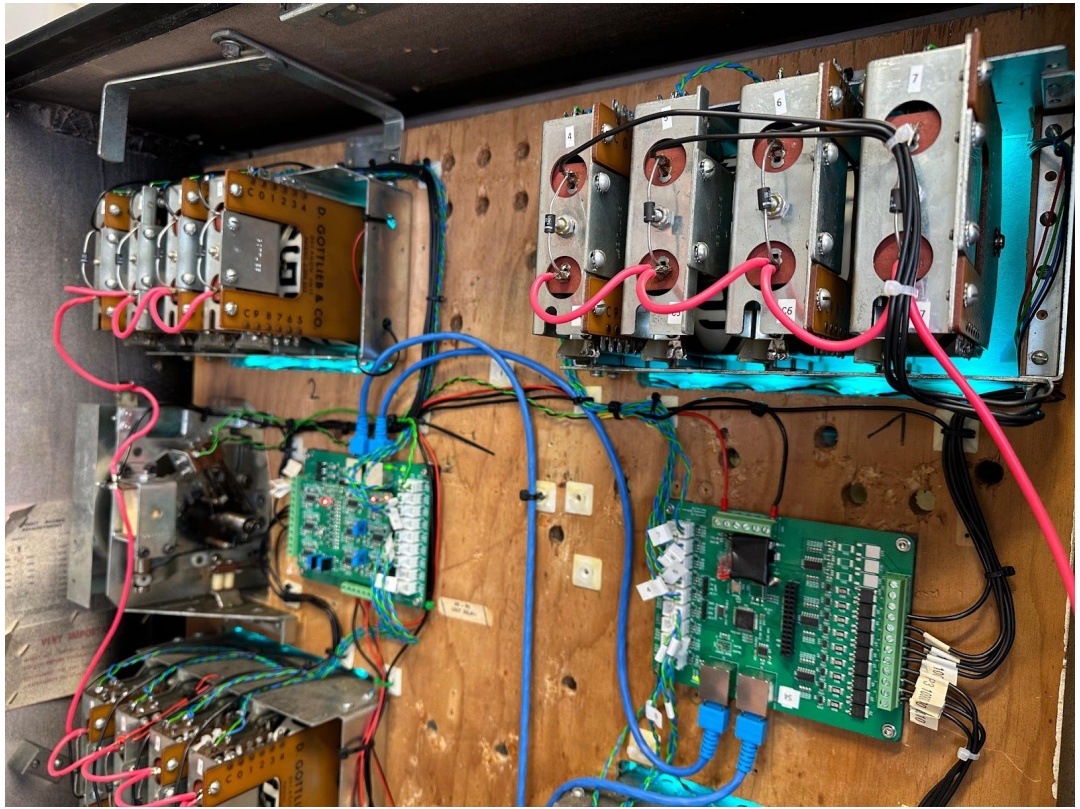


Actuators



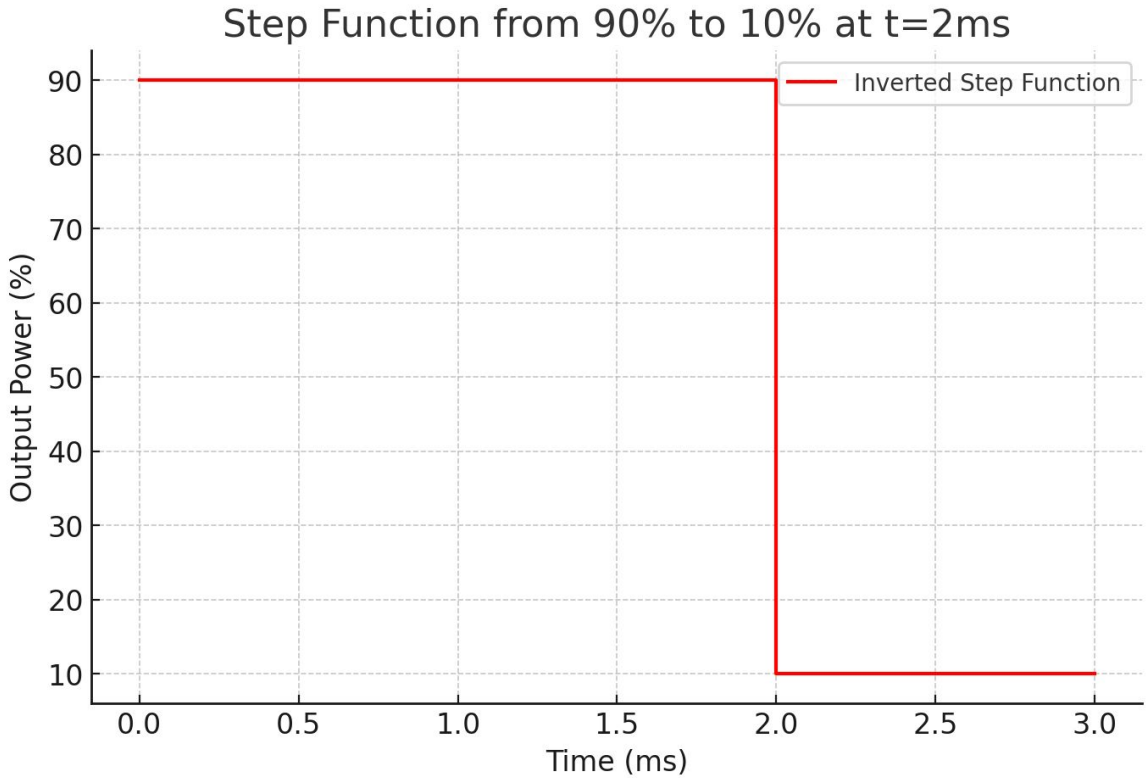
Actuators

Counters

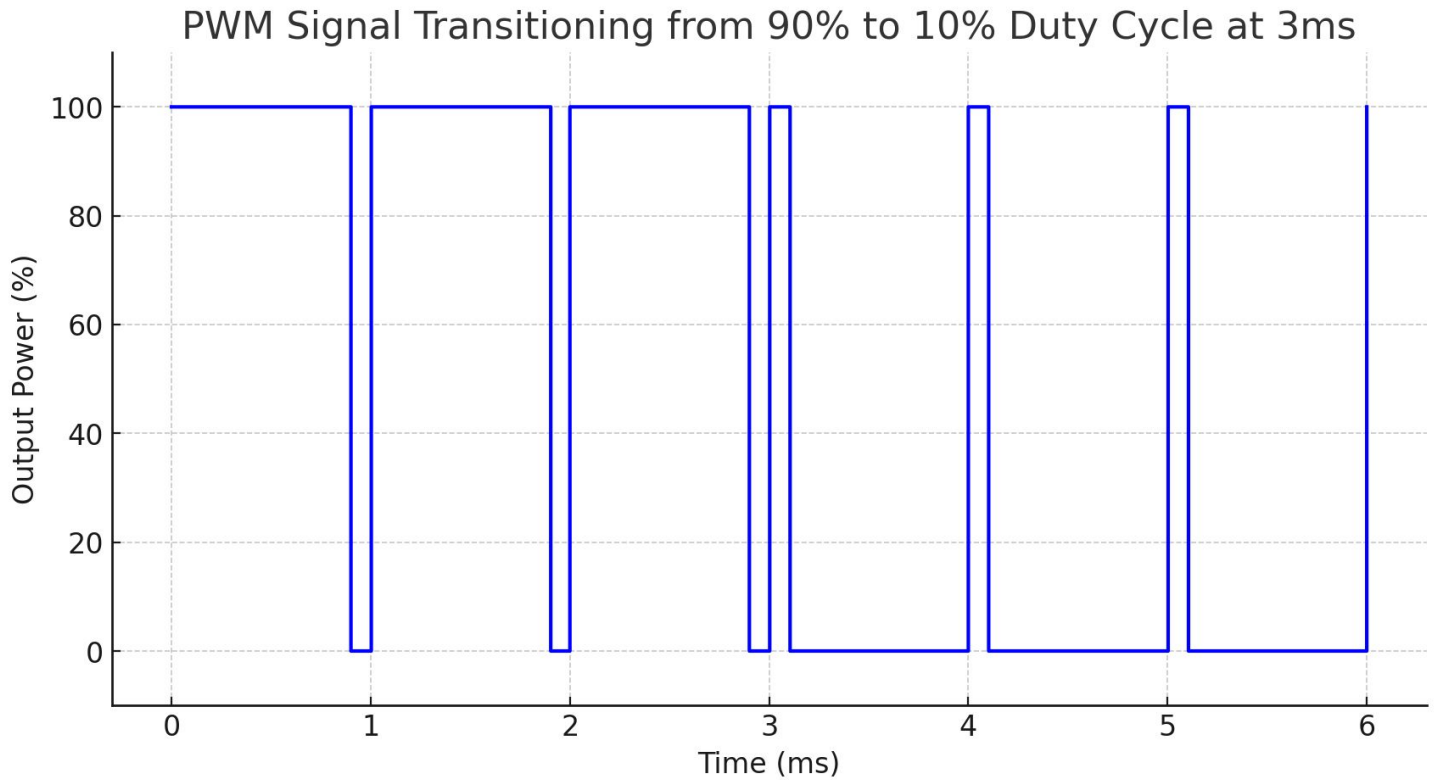


Actuators, takeaway

Actuators, technical



Actuators PWM

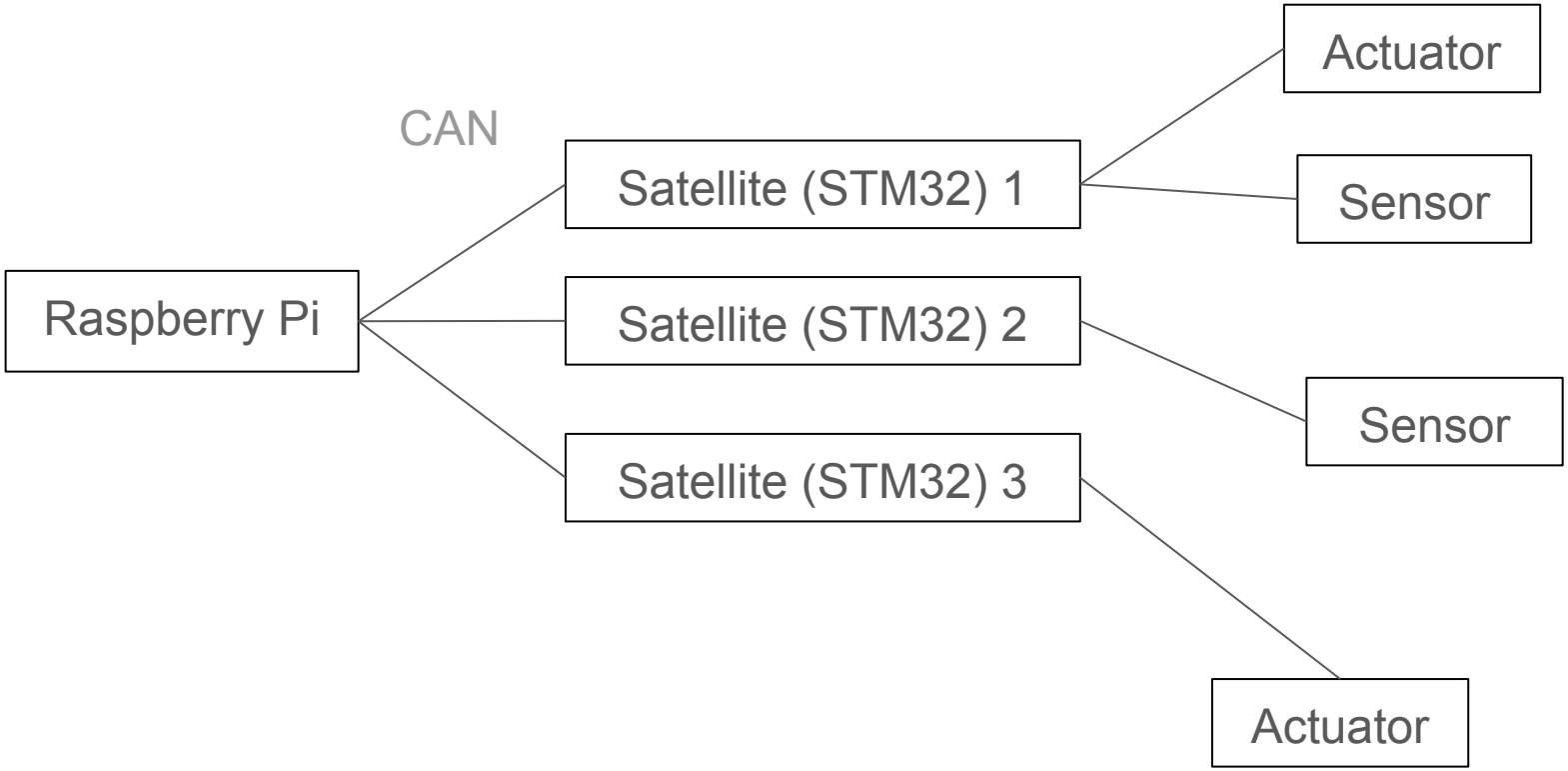


Implementation

```
analogWrite(PIN_OUTPUT, 900);
delay(3);
analogWrite(PIN_OUTPUT, 100);
```

→ 5 different codebases

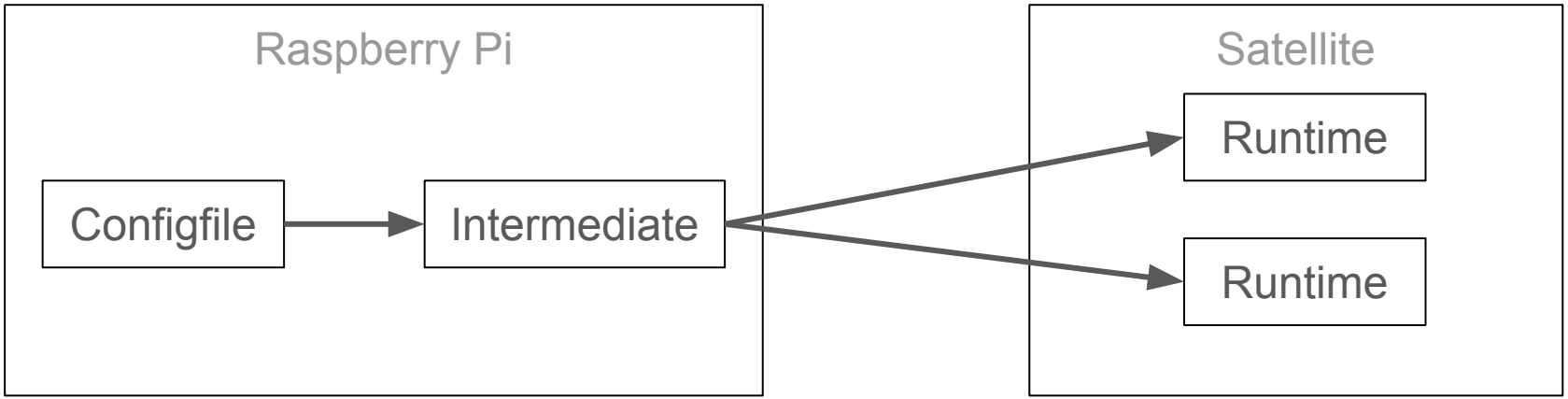
Infrastructure



Idea

1. Satellites programmed once
2. Single codebase
3. ~~Code~~ configuration
4. Config held centrally
5. Config updateable

Idea, low level control



CAN bus

- Differential pair
- Multi-master
- Collision detection/avoidance
- Short packets

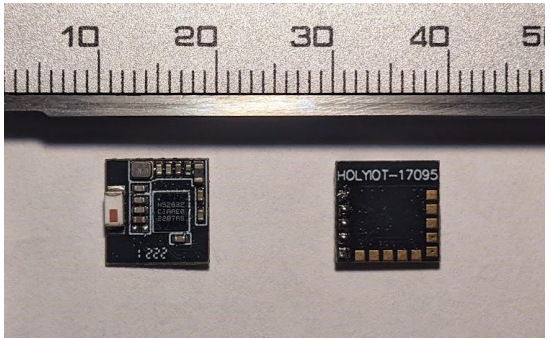
CANopen

- Object dictionary
  - Collection of “variables”
- Profiles
  - CIA 401: “Generic I/O device profile”
  - Our own: intermediary data
- Heartbeats
- …

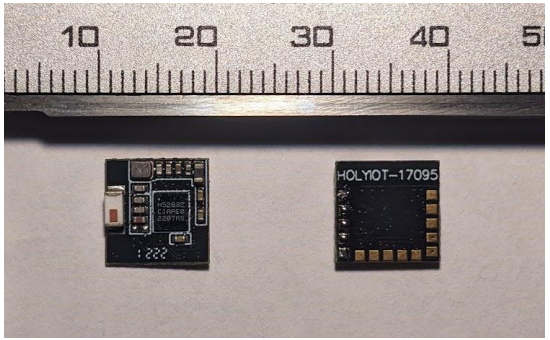
Item configuration

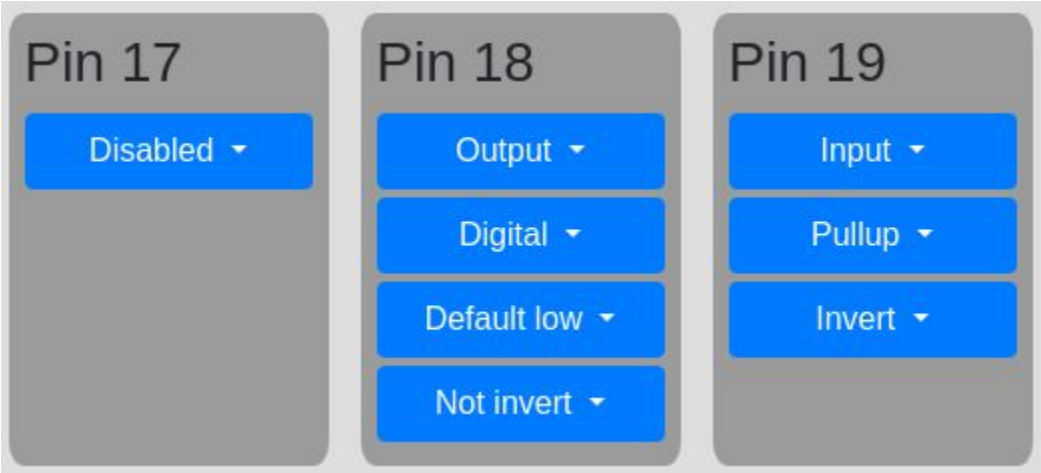
```
flippers:
  id: flipper_left_outer
  type: microcode
  code: solenoid
  duty_cycles: [[80, 30], [20]]
  analog_channel: 0
  position:
    x: 0.29
    y: 0.31
```

BLEnky

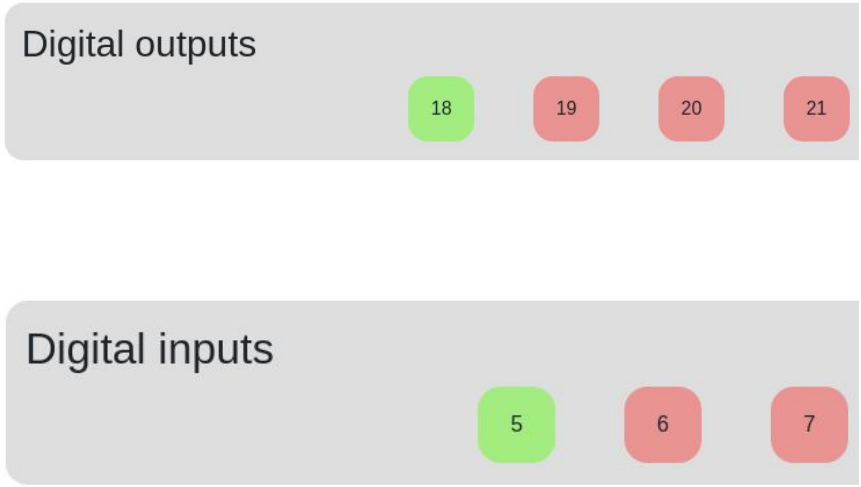


BLEnky

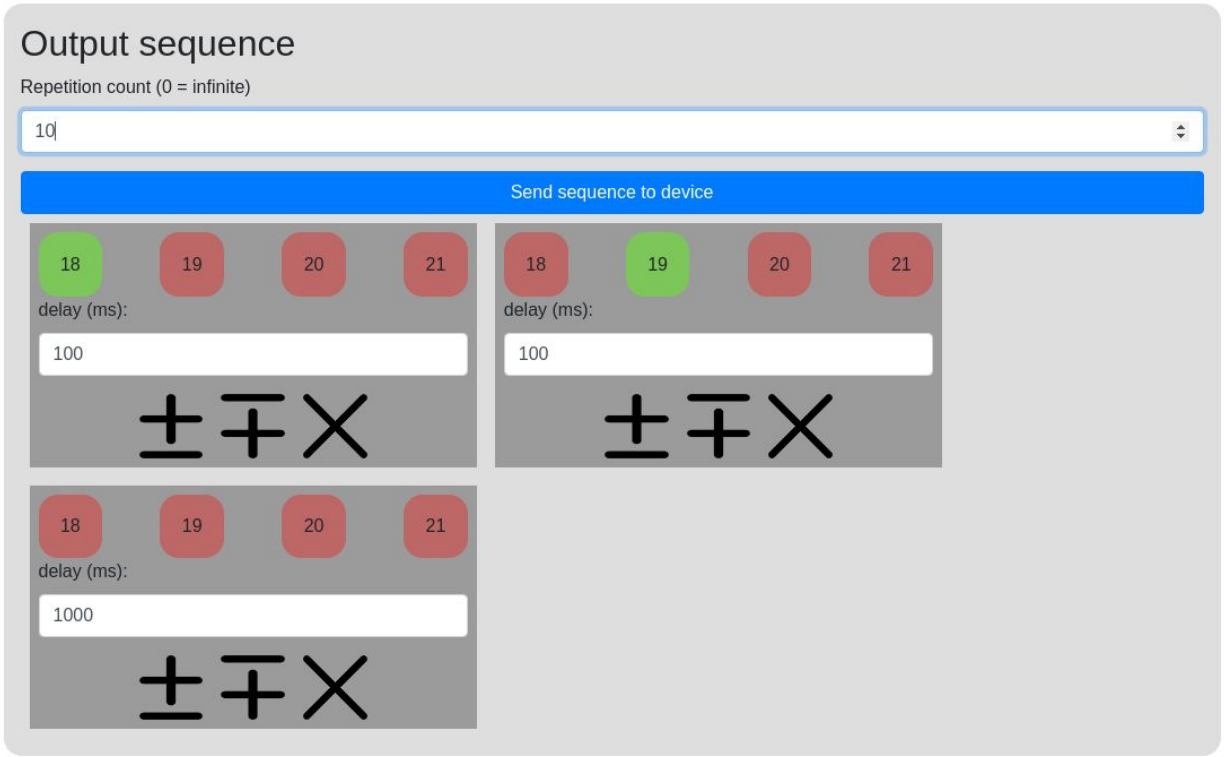

10
20
30
40
HOLYIOT-17095


Pin 17
Disabled
Pin 18
Output
Digital
Default low
Not invert
Pin 19
Input
Pullup
Invert

BLEnky the second



BLEnky, again



gpioASM

```
1     label start
2
3     write_digital 11111
4     sleep_ms 500
5     write_digital 00000
6     sleep_ms 500
7
8     jump start
```

gpioASM

```
1     label start
2
3     write_digital 11111
4     sleep_ms 500
5     write_digital 00000
6     sleep_ms 500
7
8     jump start
```

More commands:

- jump_match label 1-0
- sleep_match -00

gpioASM generation

```
flippers:
  id: flipper_left_outer
  type: microcode
  code: solenoid
  duty_cycles: [[80, 30], [20]]
  analog_channel: 0
  position:
    x: 0.29
    y: 0.31
  can:
    id: 4

```

gpioASM generation

```yaml
flippers:
  id: flipper_left_outer
  type: microcode
  code: solenoid
  duty_cycles: [[80, 30], [20]]
  analog_channel: 0
  position:
    x: 0.29
    y: 0.31
  can:
    id: 4

```

```
label start

{% if check_input %}
jump_match_all exit {{ '-' * gpio.index }}{{ (gpio.state + 1) % 2 }}
{% endif %}

{% if wait_for_input %}
sleep_match_all {{ '-' * gpio.index }}{{ gpio.state }}
{% endif %}

{% for duty_cycle in duty_cycles %}

{% if analog_channel is defined %}
write_analog_channel_{{ analog_channel }} {{ duty_cycle[0] }}
{% endif %}
```

gpioASM generation

```
flippers:
  id: flipper_left_outer
  type: microcode
  code: solenoid
  duty_cycles: [[80, 30], [20]]
  analog_channel: 0
  position:
    x: 0.29
    y: 0.31
  can:
    id: 4

```

```
label start

{% if check_input %}
jump_match_all exit {{ '-' * gpio.index }}{{ (gpio.state + 1) % 2 }}
{% endif %}

{% if wait_for_input %}
sleep_match_all {{ '-' * gpio.index }}{{ gpio.state }}
{% endif %}

{% for duty_cycle in duty_cycles %}

{% if analog_channel is defined %}
write_analog_channel {{ analog_channel }} {{ duty_cycle[0] }}
{% endif %}
```

gpioASM infrastructure

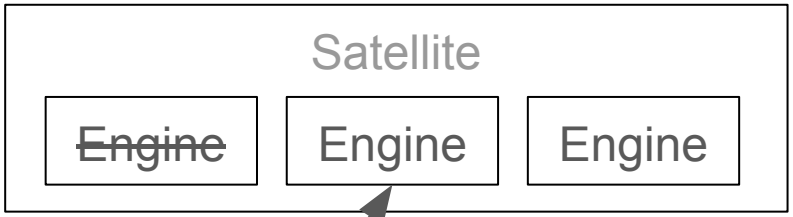
```
1    label start
2    write_analog_channel_0 80
3    sleep_ms 30
4    write_analog_channel_0 20
5    label exit
6    exit
```

gpioASM infrastructure

```
1    label start
2    write_analog_channel_0 80
3    sleep_ms 30
4    write_analog_channel_0 20
5    label exit
6    exit
```

gpioASM infrastructure

```
1    label start
2    write_analog_channel_0 80
3    sleep_ms 30
4    write_analog_channel_0 20
5    label exit
6    exit
```



DBus interface

ItemRegistry

`org.freedesktop.DBus.Properties`

Get
Gets an items state

DBus interface

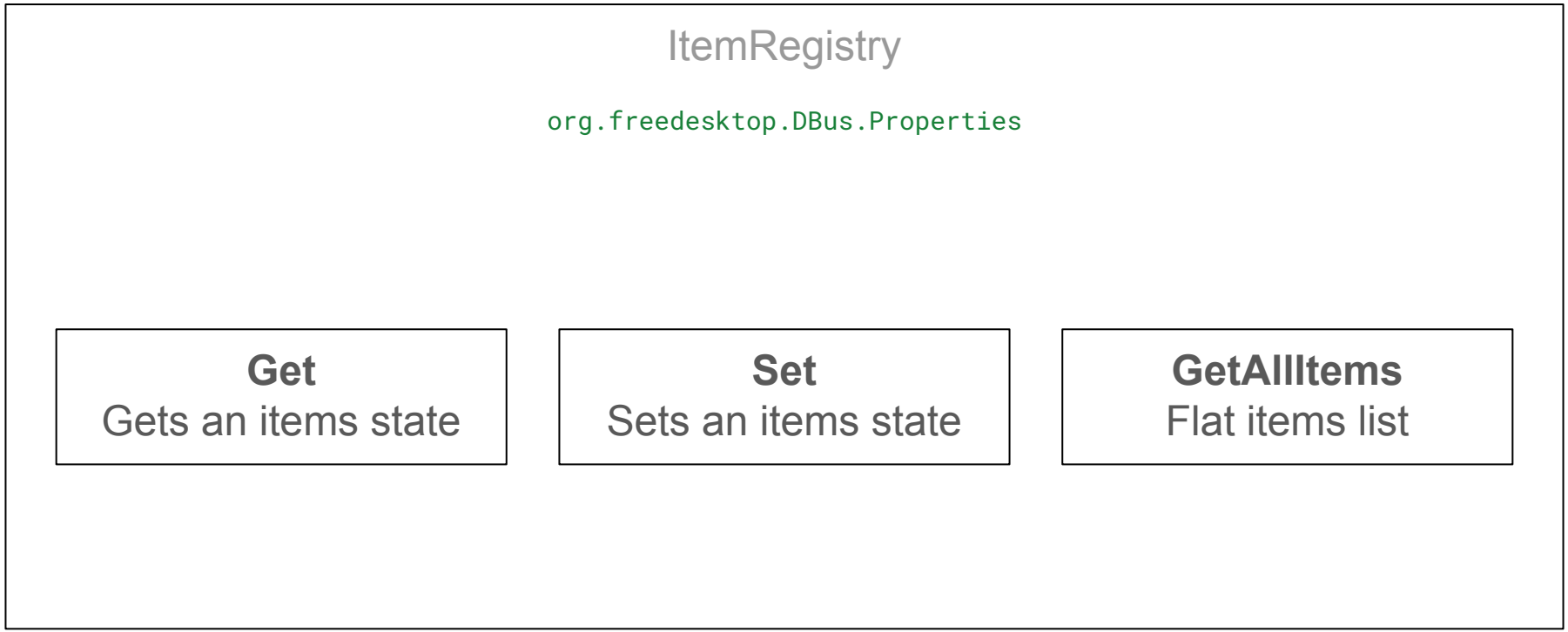
ItemRegistry

`org.freedesktop.DBus.Properties`

Get
Gets an items state

Set
Sets an items state

DBus interface



Python gamecore

```
@engine.on_item_state_changed('button_left')
def handle(item_id, item_state):
    engine.item_proxy.flipper_left_outer = item_state
    engine.item_proxy.flipper_left_inner = item_state
```

Python gamecore

```
@engine.on_item_state_changed('button_left')
def handle(item_id, item_state):
    engine.item_proxy.flipper_left_outer = item_state
    engine.item_proxy.flipper_left_inner = item_state
```

```
engine.item_follow(
    'flipper_right_outer',
    'flipper_right_inner',
    item_source='button_right'
)
```

Website

Frontend

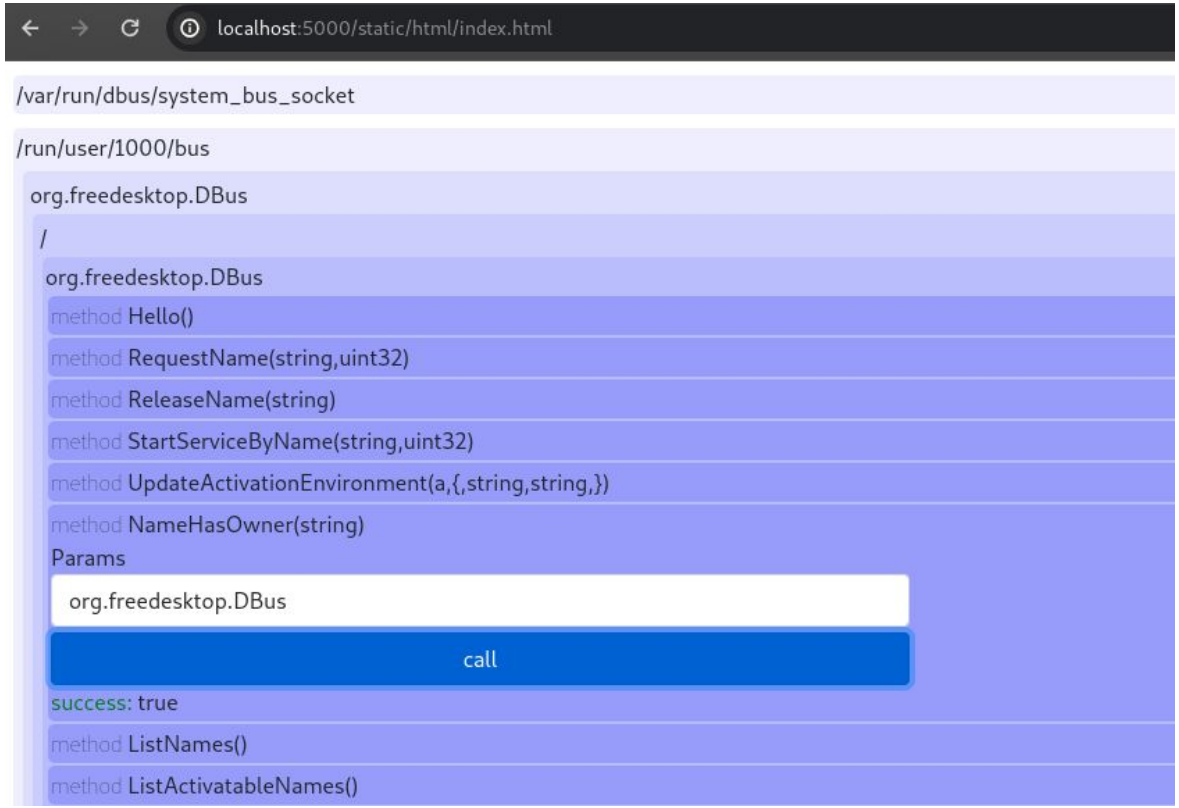
↕

HTTP-DBus gateway

↕

ItemRegistry

DBus ↔ HTTP gateway



Result

- Robust
- Flexible
- Latency of around 30ms
- Reusable
- Lots of engineering

Outlook

- Idea validated
  - Centralized control of decentralized infrastructure
  - Configuration / intermediate

Outlook

- Idea validated
  - Centralized control of decentralized infrastructure
  - Configuration / intermediate

- BLEnky
  - Platform agnostic (beyond Nordic)
  - Transport agnostic (beyond BLE)
  - Protocol standardization (Packet based)

Sources

- https://pinball-factory.com/products/king-rock-flipper
- https://www.deko-design-klein-shop.de/185317-Glockenspiel-Chime-Unit-fuer-WILLIAMS-Flipper-anschlussfertig

Portfolio

- Projects
- Presentations (incl. this one)
- Guides
- Failures

Questions?