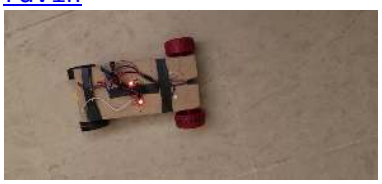
🐬 bluer-ugv

🐬 `@ugv` is a bluer-ai plugin for UGVs.

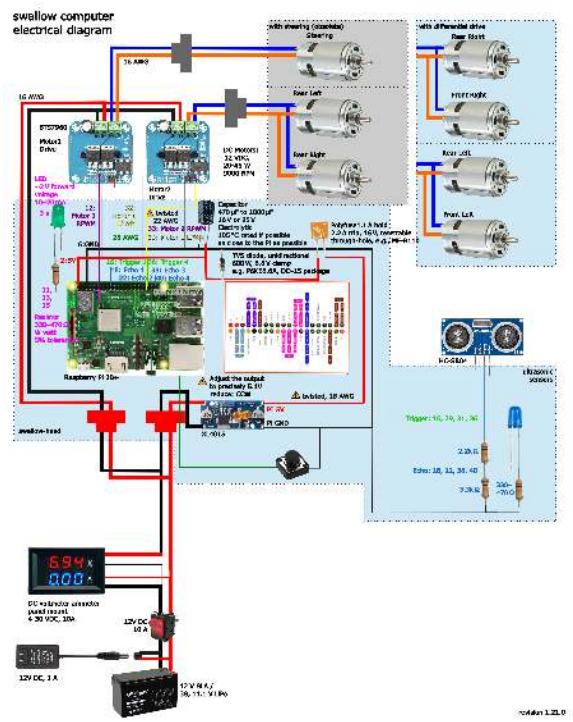
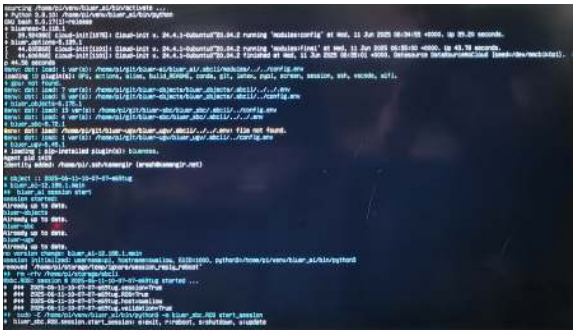
```
pip install bluer_ugv
```

designs

swallow  based on power wheels.	arzhang  swallow's little sister.	rangin  swallow's ad robot.
ravin  remote control car kit for teenagers.	eagle  a remotely controlled ballon.	fire  based on a used car.

beast  based on UGV Beast PI ROS2.		

shortcuts

computer 	UGVs 
terraform 	validations 

aliases

@swallow, @ugv.

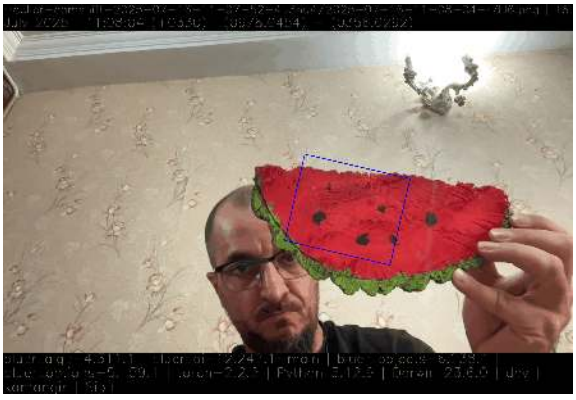
🌀 blue-rover for the Global South.

built by 🌀 bluer README, based on 🐬 bluer_ugv-7.485.1.

🪄 bluer-algo

🪄 `@algo` carries AI algo.

```
pip install bluer-algo
```

bps  bluer-positioning system.	yolo  a yolo interface.
tracker  a visual tracker.	image_classifier  an image classifier.

aliases

@bps, @image_classifier, @tracker, @yolo.

---

for the Global South.

built by 🌀 `bluer README`, based on 🪄 `bluer_algo-4.602.1`.

bluer-sbc

`bluer-sbc` is a `bluer-ai` plugin for edge computing on single board computers.

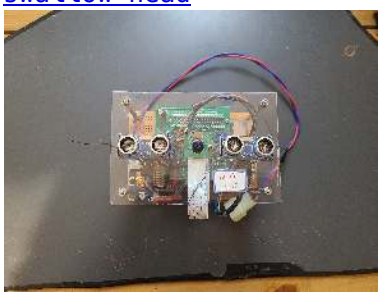
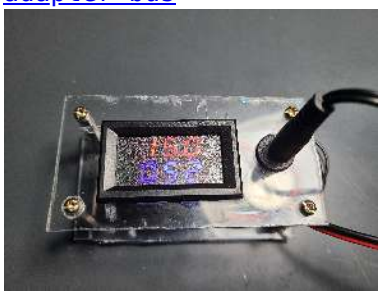
installation

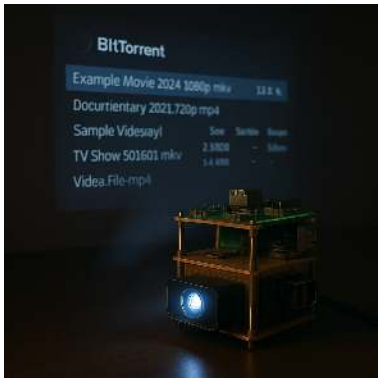
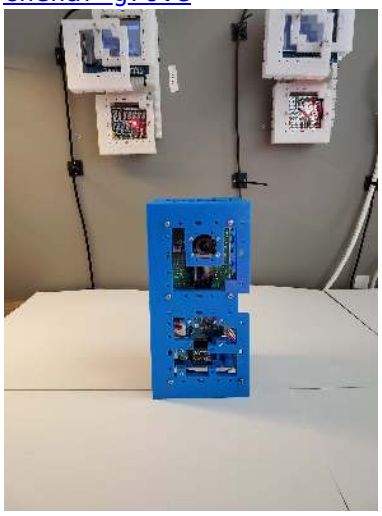
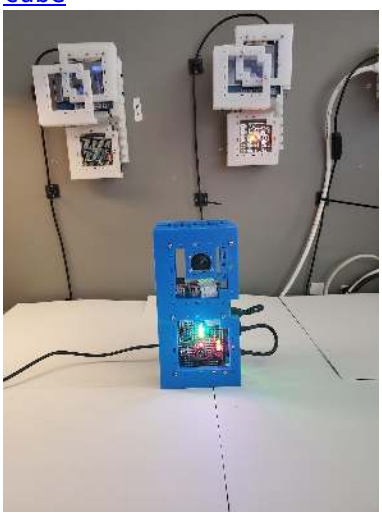
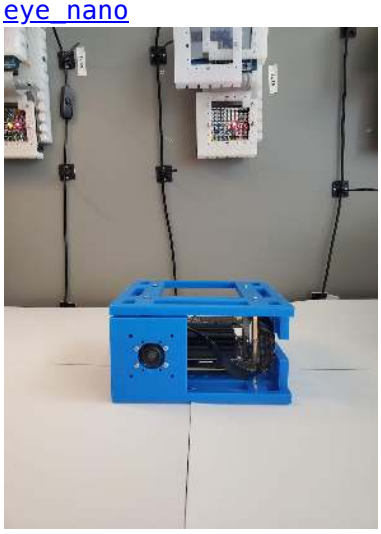
```
pip install bluer_sbc

# @env dot list
@env dot cp <env-name> local
```

aliases

@camera, @sbc rpi, @sbc .

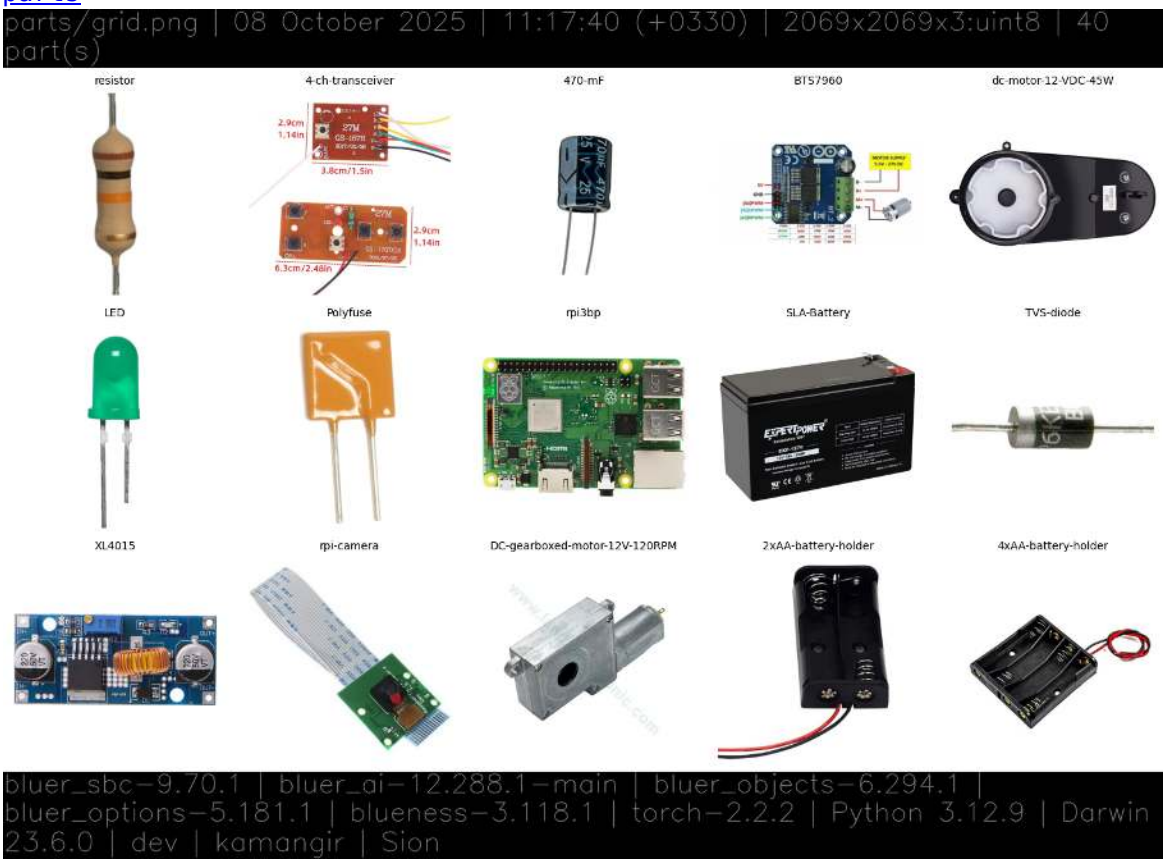
designs

swallow_head 	swallow 	pwm_generator 
regulated_bus 	battery_bus 	adapter_bus 

ultrasonic-sensor-tester 	bryce 	cheshmak 
nafha 	shelter 	blue3 
chenar-grove 	cube 	eye_nano 

shortcuts

parts



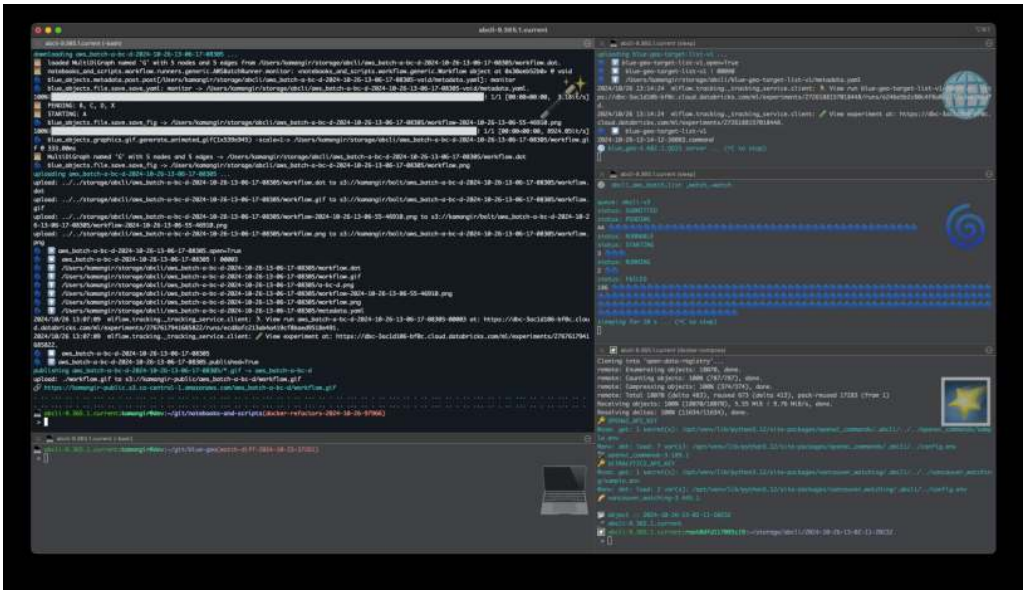
---

blue-sbc for the Global South.

built by bluer README, based on bluer_sbc-9.246.1.

bluer-ai

`bluer-ai` is an implementation of 🔻 giza and a language to speak AI.



image

installation

```
pip install bluer_ai
```

Add to `bashrc`, `.bash_profile`, or the relevant startup file.

```
source $(python3 -m bluer_ai locate)/.abcli/bluer_ai.sh
```

dev install

- Amazon EC2 instances
- Amazon SageMaker
- Jetson Nano
- macOS
- Raspberry Pi
- Raspberry Pi + ROS

aliases

@conda, @git, @gpu, @init, @latex, @logging, @plugins, @pypi, @random, @screen, @seed 🌱 , @ssh, @terraform, @today, @wifi.

@options: @assert, @badge, @browse, @cat, @code, @env, @eval, @help, @hr, @list, @not, @open, @option, @pause, @pylint, @pytest, @repeat, @sleep, @test, @timestamp, @wait, @watch.

@objects: @select, @storage.

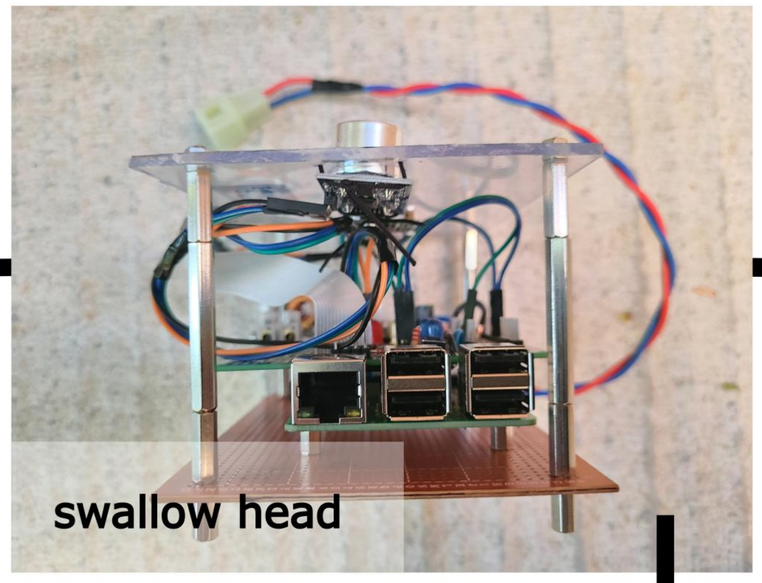
🪄 `abcli` for the Global South.

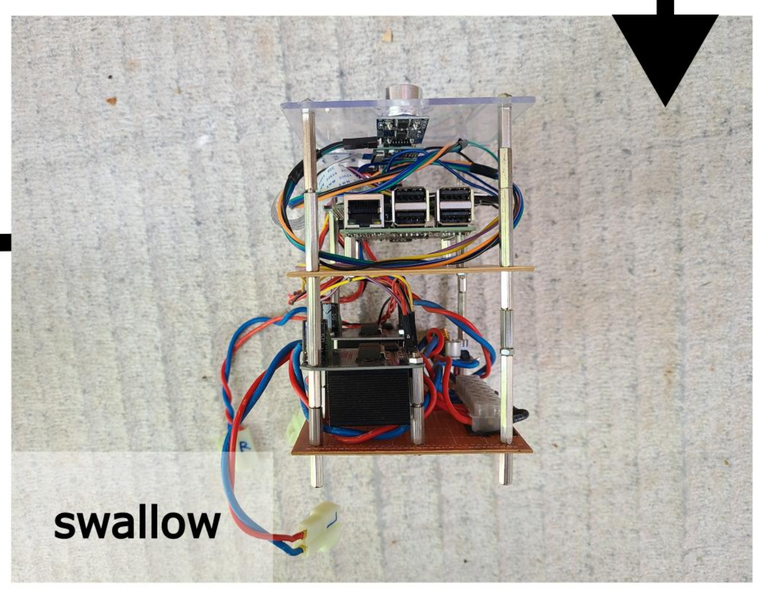
built by 🌀 `bluer README`, based on 🌀 `bluer_ai-12.332.1`.

swallow class of UGVs

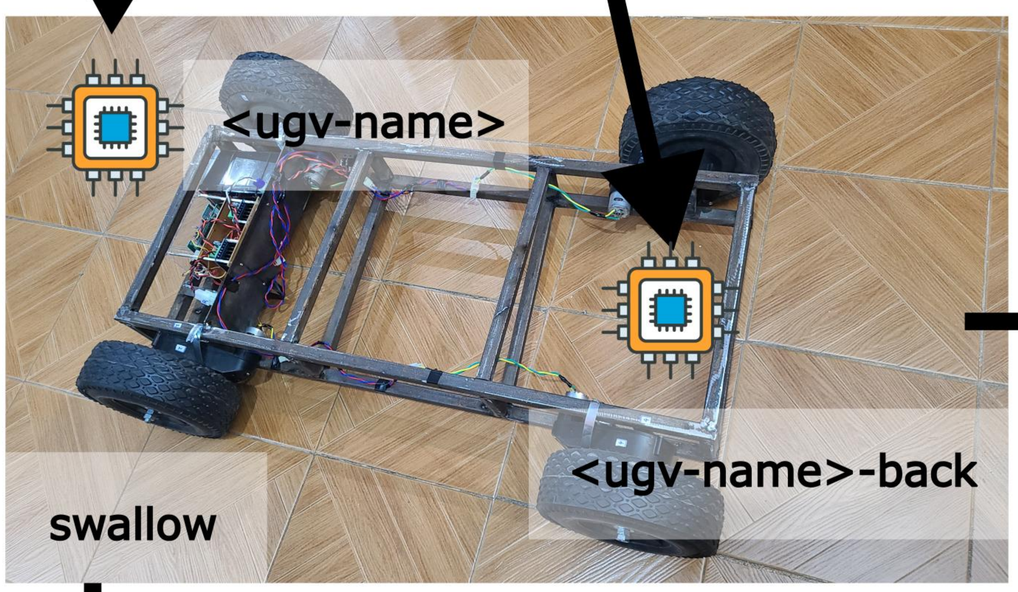
terminology: computers, designs, and UGVs

computers





designs



UGVs

swallow

arzhang

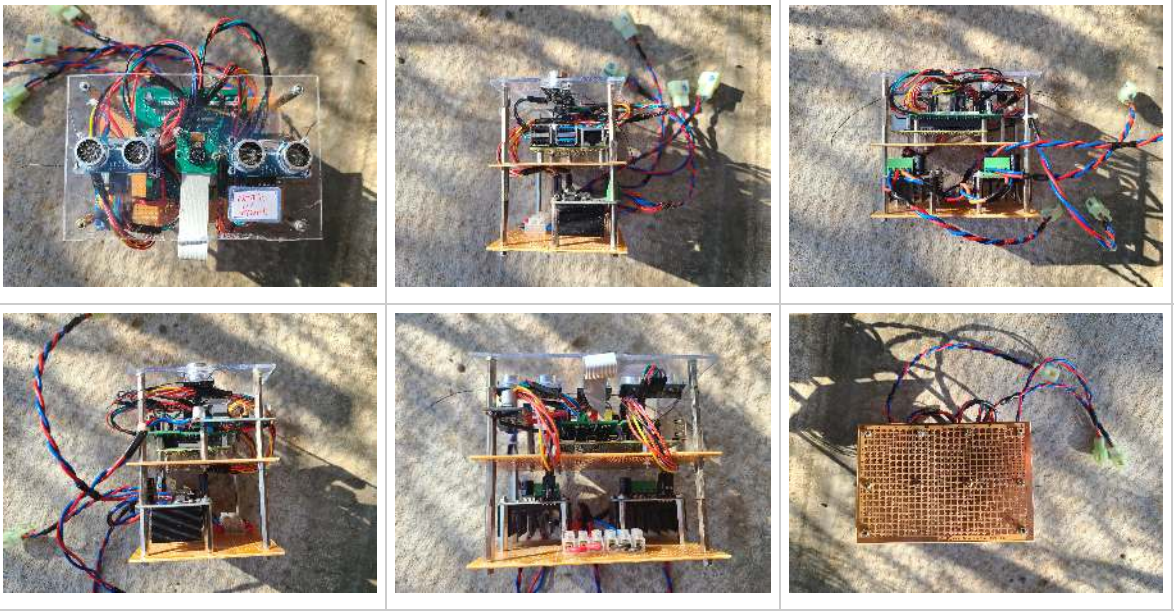
arzhang2

arzhang3

rangin

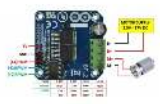
swallow

- minimum hardware for UGV control.
- Raspberry Pi, 5 MP camera, 2+2 ultrasonic sensors, 2 x 12 VDC 43A DC motor drivers.
- width x height x length: 140 mm x 95 mm x 130 mm
- terraform.
- previous versions: v1, v2, v3, v4, v5.



parts

swallow-head + ⬇️

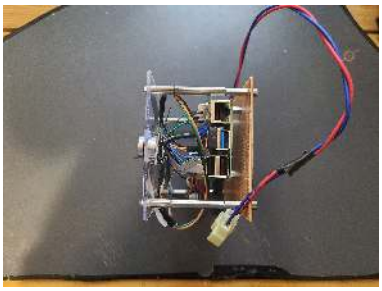
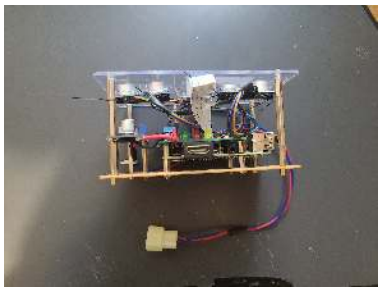
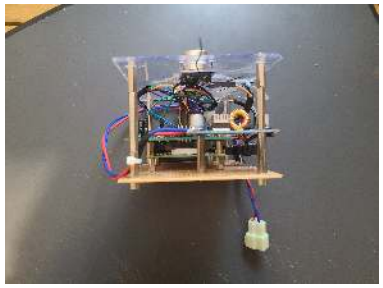
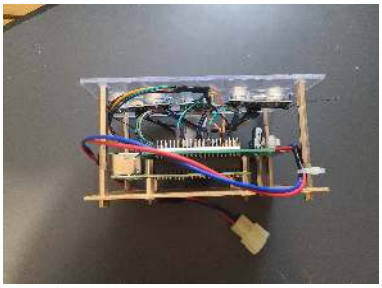
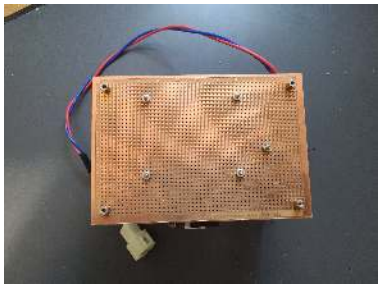
43 A, H-Bridge Motor Driver	auto power connectors	nuts, bolts, and spacers	single-sided PCB, 14 cm x 9.5 cm	solid cable 1-1.5 mm^2	white terminal
					
2 x	2 females	M3: (4 x nut + 8 x 25 mm spacer + 4 x 30 mm spacer)		20 cm x (red + black/blue)	8 x

1. 43 A, H-Bridge Motor Driver: 2 x.
2. auto power connectors: 2 females.

3. nuts, bolts, and spacers: M3: (4 x nut + 8 x 25 mm spacer + 4 x 30 mm spacer).
4. single-sided PCB, 14 cm x 9.5 cm.
5. solid cable 1-1.5 mm^2: 20 cm x (red + black/blue).
6. white terminal: 8 x.

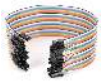
swallow-head

- head of swallow computer, without the DC motor drivers.
- width x height x length: 140 mm x 95 mm x 75 mm
- used for swallow design backs and rangin tops.
- previous versions: v1.

parts

16 AWG wire  40 cm x (red + black/blue)	HC-SR04: ultrasonic-sensor  4 x	LED, ~2 V forward voltage, 10-20 mA  green + red + yellow + 4 x blue	Polyfuse, 1.1 A hold, 2.2 A trip, 16 V, resettable, through-hole, e.g., MF-R110  optional	Raspberry Pi Camera, V1.3https://www.raspberrypi.com/documentation/accessories/camera.html 	Raspberry Pi. 	Resistor, 1/4 watt, 5% tolerance  7 x 330-470 Ω + 4 x 2.2 kΩ + 4 x 3.3 kΩ	SD card, 32 GB 	TVS diode, unidirectional, 600 W, 6.8 V clamp, e.g. P6KE6.8A, DO-15 package 	XL4015: 8 - 36 VDC -> 1.25 - 32 VDC, 5A 

auto power connectors	capacitor, 470 µF to 1000 µF, 16 V or 25 V, Electrolytic, 105 °C rated if possible.	dupont cables, female to female	green terminal	nuts, bolts, and spacers	pin headers	plexiglass, 2 mm or 2.5 mm thickness	push button	single-sided PCB, 14 cm x 9.5 cm	solid cable 1-1.5 mm^2
 1 female		 1 x 30 cm + 1 x 10 cm	 2 x	 M2: (2 x bolt + 4 x nut + 2 x 5 mm spacer) + M2.5: (4 x bolt + 4 x nut + 8 x 10 mm spacer) + M3: (1 x bolt + 5 x nut + 5 x 25 mm spacer + 8 x 15 mm spacer + 4 x 5 mm spacer)	 1 x (female, 2 x 40) -> 2 x 20 + 2 x (male, 1 x 40) -> 4 x 1 + 2 x 20 + 1 x (male, 2 x 40) -> 2 x 2 x 6	14 cm x 9.5 cm			 10 cm x (red + black/blue)
strong thread  1 m									

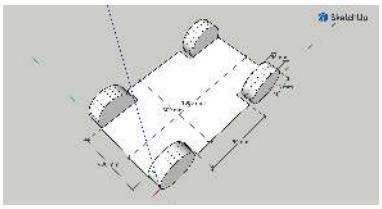
1. 16 AWG wire: 40 cm x (red + black/blue).
2. HC-SR04: ultrasonic-sensor: 4 x.
3. LED, ~2 V forward voltage, 10-20 mA: green + red + yellow + 4 x blue.
4. Polyfuse, 1.1 A hold, 2.2 A trip, 16 V, resettable, through-hole, e.g., MF-R110: optional.
5. Raspberry Pi Camera, V1.3https://www.raspberrypi.com/documentation/accessories/camera.html.
6. Raspberry Pi..

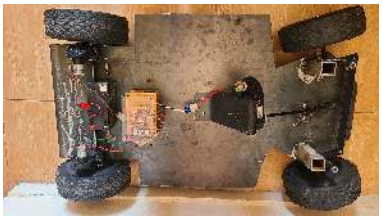
7. Resistor, 1/4 watt, 5% tolerance: 7 x 330-470 Ω + 4 x 2.2 kΩ + 4 x 3.3 kΩ.
8. SD card, 32 GB.
9. TVS diode, unidirectional, 600 W, 6.8 V clamp, e.g. P6KE6.8A, DO-15 package.
10. XL4015: 8 - 36 VDC -> 1.25 - 32 VDC, 5A.
11. auto power connectors: 1 female.
12. capacitor, 470 µF to 1000 µF, 16 V or 25 V, Electrolytic, 105 °C rated if possible..
13. dupont cables, female to female: 1 x 30 cm + 1 x 10 cm.
14. green terminal: 2 x.
15. nuts, bolts, and spacers: M2: (2 x bolt + 4 x nut + 2 x 5 mm spacer) + M2.5: (4 x bolt + 4 x nut + 8 x 10 mm spacer) + M3: (1 x bolt + 5 x nut + 5 x 25 mm spacer + 8 x 15 mm spacer + 4 x 5 mm spacer).
16. pin headers: 1 x (female, 2 x 40) -> 2 x 20 + 2 x (male, 1 x 40) -> 4 x 1 + 2 x 20 + 1 x (male, 2 x 40) -> 2 x 2 x 6.
17. plexiglass, 2 mm or 2.5 mm thickness: 14 cm x 9.5 cm.
18. push button.
19. single-sided PCB, 14 cm x 9.5 cm.
20. solid cable 1-1.5 mm^2: 10 cm x (red + black/blue).
21. strong thread: 1 m.

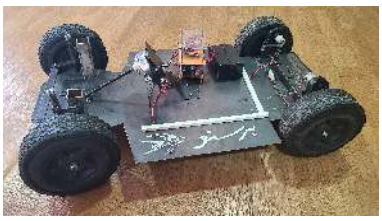
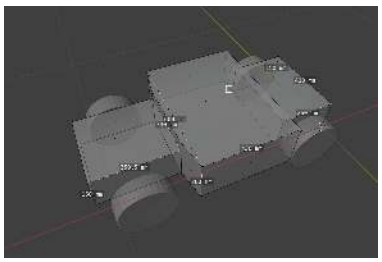
swallow

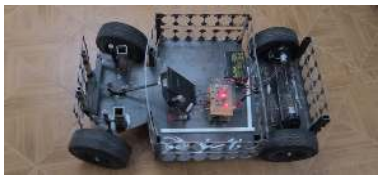
`swallow` is a family of UGVs named after the Iranian film “The Swallows Return to Their Nest” (پرستوها به لانه برمی‌گردند) from the 1940s (wikipedia), (irmdb), (pictures), (pictures). arzhang and rangin belong to this family.

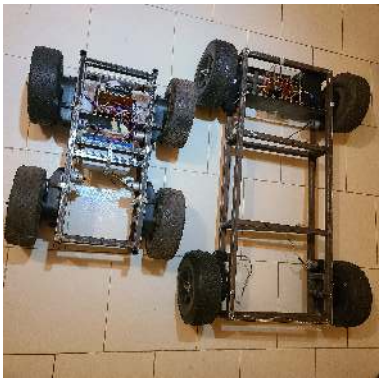
- analog control
- digital control

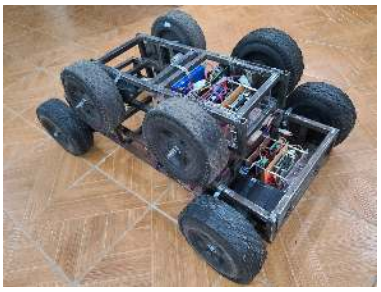



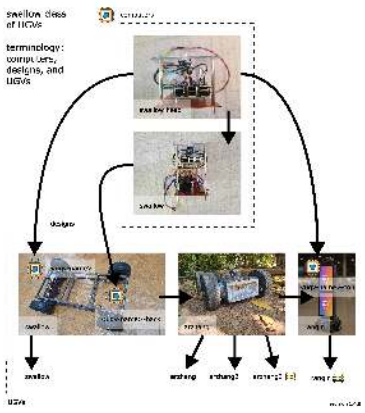






terminology:
computers,
designs, and
UGVs

aliases: swallow

dataset

```
@swallow \
    dataset \
    combine \
    [count=<count>,~download,~recent,sequence=<3>,~split,upload] \
    [-|<object-name>] \
    [--datasets <object-name-1>+<object-name-2>] \
    [--test_ratio 0.1] \
    [--train_ratio 0.8]
 . combine swallow datasets.
@swallow \
    dataset \
    download \
    [~metadata,navigation|yolo]
 . download the swallow dataset.
@swallow \
    dataset \
    edit \
    [~download,navigation|yolo]
 . edit the swallow dataset.
@swallow \
    dataset \
    list \
    [~download,navigation|yolo]
 . list the swallow dataset.
@swallow \
    dataset \
    upload \
    [~metadata,navigation|yolo]
 . upload the swallow dataset.
```

debug

```
@swallow \
    debug \
    [~upload] \
    [-|<object-name>] \
    [--generate_gif 0] \
    [--save_images 0]
 . debug swallow.
```

env

```
@swallow \
    env \
    cp \
    [<env-name>]
 . cp swallow swallow-raspbian-<env-name>.env.
@swallow \
    env \
    list
```

```
 . list swallow envs.
@swallow \
    env \
    set \
    bps | full_keyboard | steering \
    0 | 1
 . set env.
   bps: BLUER_SBC_SWALLOW_HAS_BPS (currently: 0)
   full_keyboard: BLUER_SBC_SWALLOW_HAS_FULL_KEYBOARD (currently: 0)
   steering: BLUER_SBC_SWALLOW_HAS_STEERING (currently: 1)
```

keyboard

```
@swallow \
    keyboard \
    test \
    [dryrun] \
    [--keys 1234567890-+/.]
 . test keyboard.
```

select-target

```
@swallow \
    select_target \
    [--host <hostname>] \
    [--loop 0]
 . select swallow target.
```

ultrasonic-sensor

```
@swallow \
    ultrasonic \
    review \
    [~download,upload] \
    [.|<object-name>] \
    [--frame_count <-1>] \
    [--gif 0] \
    [--rm_blank 0]
 . review ultrasonic sensor data.
@swallow \
    ultrasonic \
    test \
    [~upload] \
    [-|<object-name>] \
    [--export 0] \
    [--frame_count <-1>] \
    [--gif 0] \
    [--log 0] \
    [--max_m 0.80] \
    [--rm_blank 0]
 . test ultrasonic sensors.
```

video

```
@swallow \
    video \
    play \
    [--dryrun 1] \
    [--download 0] \
```

```
    [--engine mpv | vlc] \
    [--loop 0] \
    [--object_name <rangin-video-list-1>] \
    [--timeout <-1 | 10>] \
    [--video <loading|1>]
 . play <object-name>/<video>.
@swallow \
    video \
    playlist \
    cat \
    [download]
 . cat swallow playlist.
@swallow \
    video \
    playlist \
    download \
    [filename=<filename>,policy=different|doesnt_exist|none]
 . download swallow playlist.
@swallow \
    video \
    playlist \
    edit \
    [download]
 . edit swallow playlist.
@swallow \
    video \
    playlist \
    upload \
    [filename=<filename>,public,zip]
 . upload swallow playlist.
```

rangin-video-list-1

```
messages:
  loading:
    filename: loading_circle_bars.mp4
    source: https://www.videezy.com/backgrounds/14052-loading-circle-bars
  warning:
    filename: vecteezy_flashing-neon-warning-text-video-good-for-danger-
sign_6299554.mp4
    source: https://www.vecteezy.com/video/6299554-flashing-neon-warning-
text-video-good-for-danger-sign-illustrations
playlist:
- filename: 731d19dc3ec2f52c626eb575d61bf19b51289493-1080p.mp4
  source: https://www.aparat.com/v/d14c1r8
- filename: d166d435a33dfd7e77b29ceaa96d2cb312692024-480p.mp4
  source: https://www.aparat.com/v/a79o81f
```

swallow: digital: design: parts

a swallow computer + an optional swallow-head computer at the back + 2 optional swallow-head computers at the top + ⬇️

1. 12 VDC motor, 20-45 W: type 2, 2 x right + 2 x left.
2. AC to DC power adapter: 12V DC, 1 A.
3. DC power plug, 5.5 mm.
4. Gearboxed DC Motor, 12 V (3-24 V), 3A, 120 RPM, 1:91, 15 Kg cm: 4 x, replacement gearboxes.
5. Rechargeable sealed lead acid battery: 12 V, 7.2 Ah.
6. auto power connectors: 4 pairs.
7. heavy duty pipe clamp: 4 x 350+ mm.
8. on/off DC switch with indicator led: 12V DC 10 A.
9. power wheel wheels: 4 x .

12 VDC motor, 20-45 W  type 2, 2 x right + 2 x left	AC to DC power adapter  12V DC, 1 A	DC power plug, 5.5 mm 
Gearboxed DC Motor, 12 V (3-24 V), 3A, 120 RPM, 1:91, 15 Kg cm  4 x, replacement gearboxes	Rechargeable sealed lead acid battery  12 V, 7.2 Ah	auto power connectors  4 pairs

heavy duty pipe clamp  4 x 350+ mm	on/off DC switch with indicator led  12V DC 10 A	power wheel wheels  4 x

swallow: digital: design: terraform

Raspbian 64-bit

> ⭐ preferred.
>
> ⚠️ on 32-bit, opencv, torch, and other modules install with challenges, and likely at lower versions.

1. follow RPi (use 64-bit + headless or not).

2. run in another terminal and paste the seed 🌱 into the ssh window.

```
@seed swallow_raspbian clipboard
```

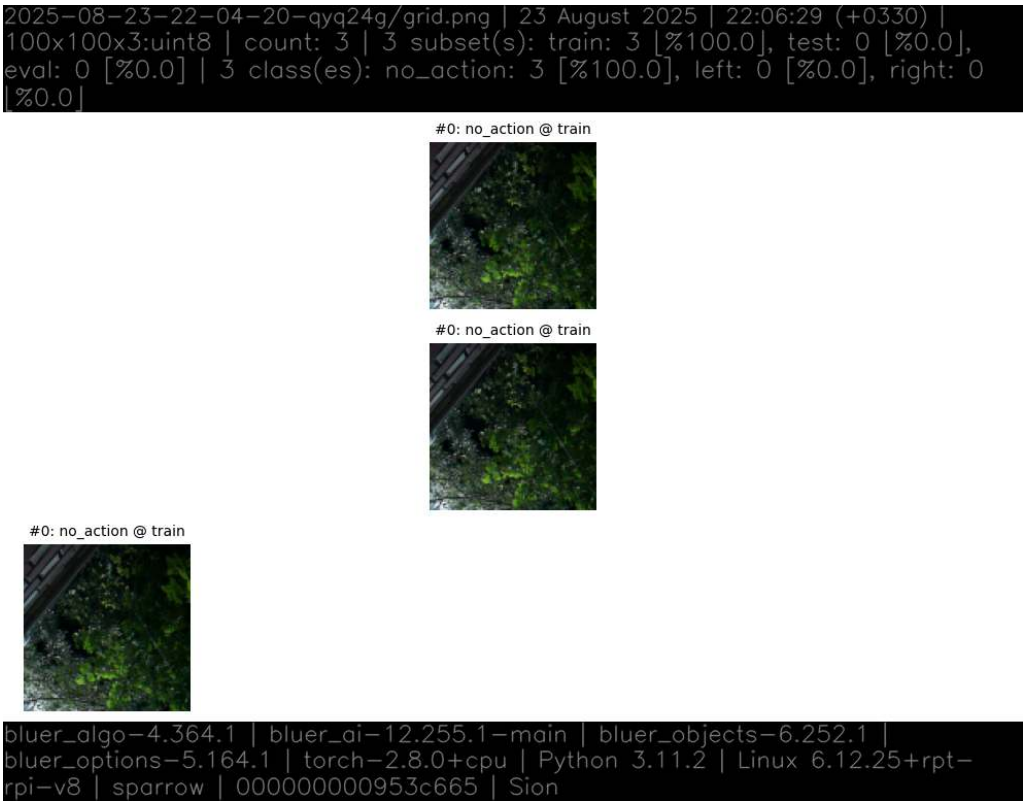
3. run,

```
@bps install
```

4. run,

```
@swallow env cp navigation
@swallow env set full_keyboard 1
@init; @select; @session start
```

now press `t`, then `w`, and wait for ~20 seconds (or press `a`, `d`), then press `i`. an dataset should be uploaded that contains a few frames from the camera.



image

5. run,

```
@swallow env cp yolo
@swallow env set full_keyboard 1
@swallow env set bps 1
@init; @select; @session start
```

6. the terraform is complete, shut down the machine,

```
@host shutdown
```

▶ Ubuntu 64-bit

Raspberry Pi

1. Open Raspberry Pi Imager and select `Operating System` -> `Raspberry Pi OS (other)` and select an image, either lite (headless) or full (with GUI).

   ℹ️ A 32-bit legacy image, such as `Bullseye`, has better compatibility with the camera, but will be more challenging when trying to install the latest OpenCV and torch. **64-bit images, are, therefore, recommended**.

2. Then press `Shift+Ctrl+X` to open `Advanced options`, select `Set hostname`, and enter a unique host name, the select `Set username and password` and set the user to `pi` and the password to `abcli2025`. Then select `Configure wireless LAN` and set the SSID and password. In `Services` select `Enable SSH` and `Use password authentication` and press `Save`. Proceed to write the SD Card.

3. After the write is complete, eject the SD card, insert it into the Raspberry Pi, turn the motherboard on, and wait for the motherboard to boot.

Headless

4. run `@seed headless_rpi clipboard` (or `headless_rpi_64_bit`). now, run `@ssh rpi <host-name>` and paste the seed 🌱 into the ssh window.

5. Run `sudo raspi-config` -> `Interfacing Options` -> enable `Camera` (only needed on older versions) and other interfaces as needed, then go to `System Options` -> `Boot / Auto Login` and select `Console Autologin`.

With UI

4. run `@seed rpi_64_bit clipboard`. now, run `@ssh rpi <host-name>` and paste the seed 🌱 into the ssh window.

5. Run `sudo nano /boot/config.txt` (or `/boot/firmware/config.txt` in later versions) and uncomment the line that reads `#hdmi_force_hotplug=1`, if it exists. If you wish to rotate the screen, add one of the following to the end of this file and then reboot:

```
display_rotate=0
display_rotate=1
display_rotate=2
display_rotate=3
```

6. go to `Preferences` -> `Raspberry Pi Configuration` -> `Interfaces` and change the following to `Enabled`: `Camera:`, `I2C`, and `SSH`, as needed.

known issues

1. on `rpi4b` headless install raises `ERROR: Wheel 'tensorflow' located at /home/pi/tensorflow-2.2.0-cp37-cp37m-linux_armv7l.whl is invalid.`

2. `pyarrow` may cause an “illegal instruction” error. this will stop plugins from loading. to recognize if this is the case, run,

```
@plugins list_of_external
```

the output should look like,

```
4 external plugin(s): bluer_algo, bluer_objects, bluer_sbc, bluer_ugv
```

if “illegal instruction” was printed, run,

```
pip uninstall -y pyarrow
```

3. rpi may reboot at the first pytorch inference, because of under-voltage. to validate, run,

```
dmesg
```

look for,

```
[   12.064503] hwmon hwmon1: Undervoltage detected!
[   16.093659] hwmon hwmon1: Voltage normalised
```

swallow computer electrical diagram

with steering (obsolete)

Steering

with differential drive

Rear Right

Front Right

16 AWG

16 AWG

Rear Left

Rear Right

BTS7960

Motor1 Drive

B-, B+, M+, M- 2

Motor2 Drive

DC Motors:
12 VDC,
20-45 W
9000 RPM

Rear Left

Front Left

LED
~2 V forward voltage
10–20 mA

3 x

12: Motor 1 RPWM

32: Motor 1 LPWM

28 AWG

6:GND

⚠ twisted 22 AWG

33: Motor 2 RPWM

35: Motor 2 LPWM

Capacitor
470 µF to 1000 µF
16 V or 25 V
Electrolytic
105 °C rated If possible
as close to the PI as possible

Polyfuse1.1 A hold
2.2 A trip, 16 V, resettable
through-hole, e.g., MF-R110

TVS diode, unidirectional
600 W, 6.8 V clamp
e.g. P6KE6.8A, DO-15 package

2:5V

16: TRigger 1 36: TRigger 4

18: Echo 1 38: Echo 3

22: Echo 2 40: Echo 4

37: push button

31: Trigger 3

29: Trigger 2

11,
13,
15

Resistor
330–470 Ω
¼ watt
5% tolerance

Raspberry PI 3B+

Adjust the output to precisely 5.1V reduce: CCW

⚠ twisted, 18 AWG

PI 5V

PI GND

XL4015

HC-SR04

ultrasonic sensors

TRigger: 16, 29, 31, 36

2.2kΩ

Echo: 18, 22, 38, 40

3.3kΩ

330–470Ω

swallow-head

6.94 V

0.00 A

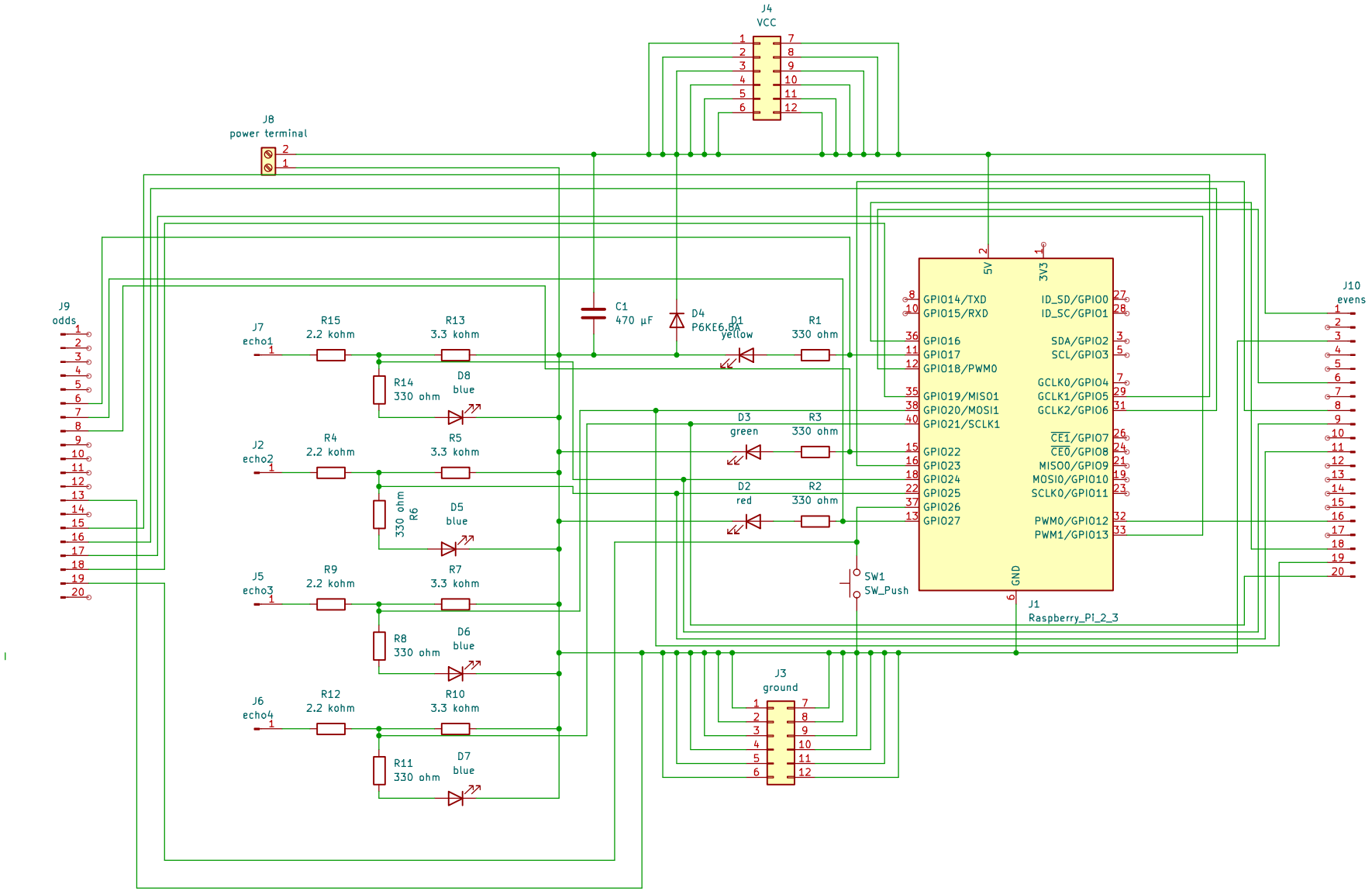
DC voltmeter ammeter
panel mount
4-30 VDC, 10A

12V DC
10 A

12V DC, 1 A

12 V SLA /
3S, 11.1 V LiPo

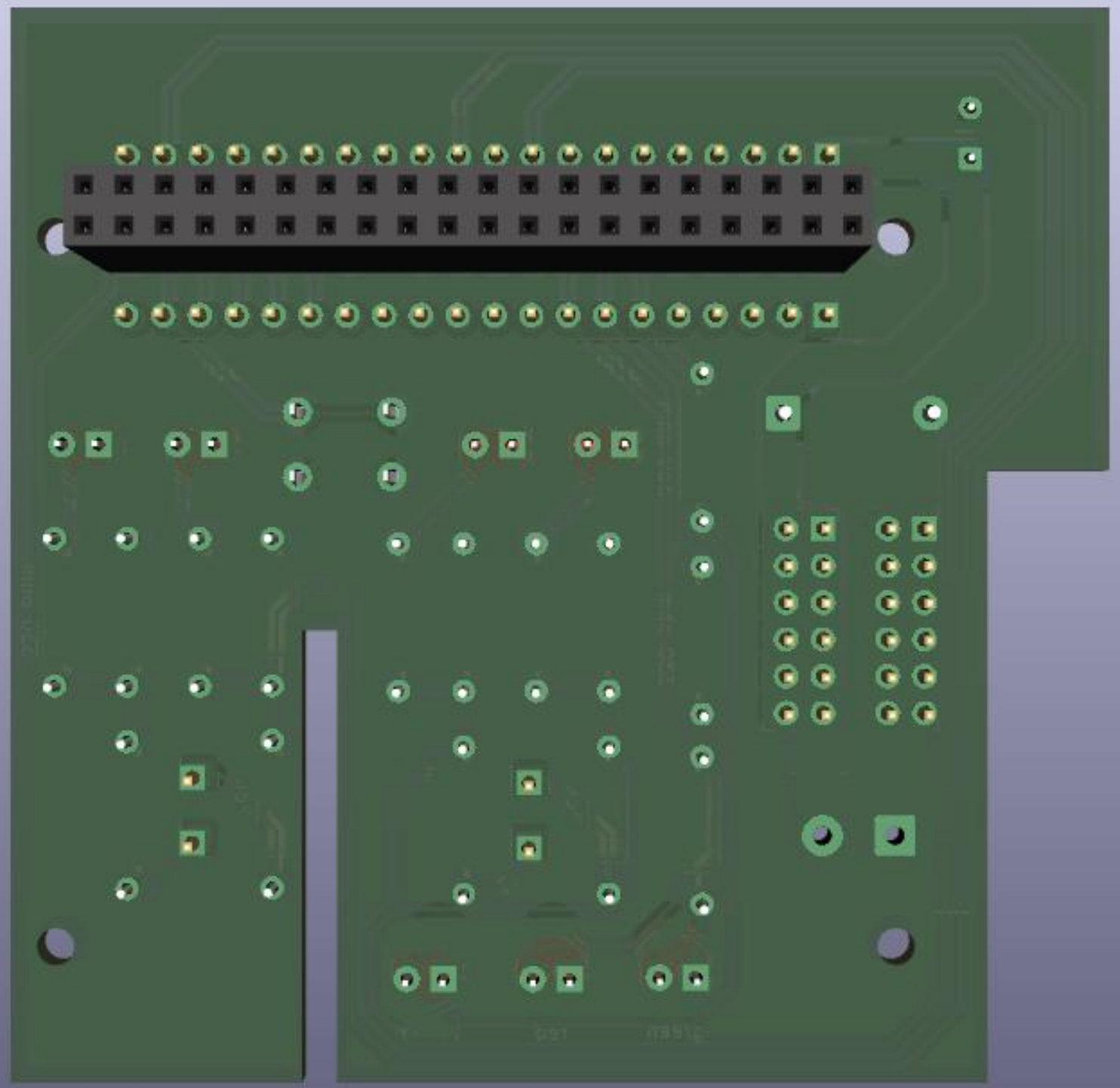
revision 1.21.0

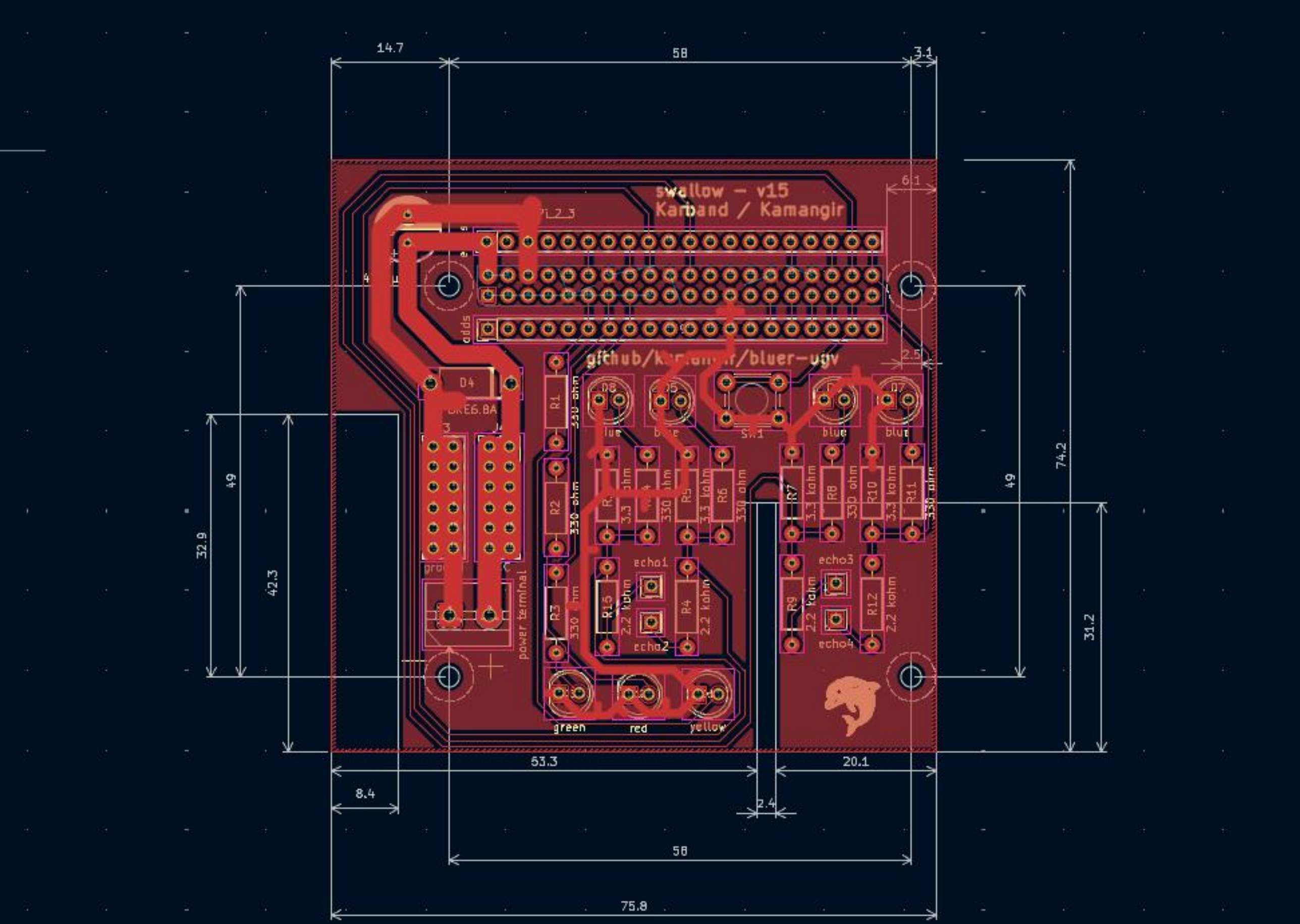


swallow shield

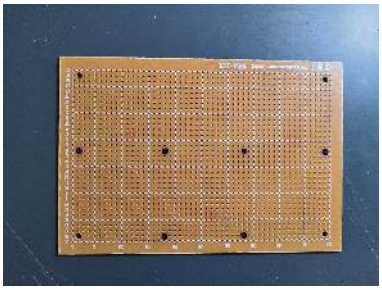
v15

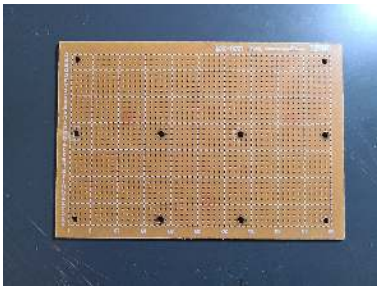
swallow – v15
Karband / Kamangir
Raspberry_PI_2_3
evens
odds
470 µF
P6KE6.8A
J3
J4
power terminal
330 ohm
github/kamangir/bluer-ugv
blue
blue
SW1
blue
blue
3.3 kohm
330 ohm
3.3 kohm
330 ohm
2.2 kohm
2.2 kohm
echo1
echo2
3.3 kohm
330 ohm
3.3 kohm
330 ohm
2.2 kohm
2.2 kohm
echo3
echo4

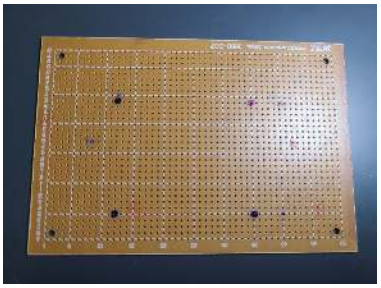


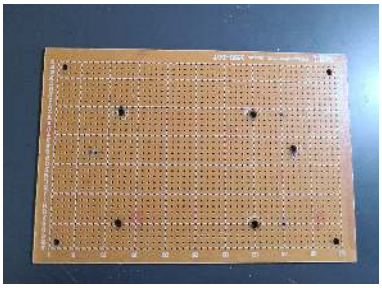

14.7
58
3.1
6.1
2.5
swallow – v15
Ka band / Kamangir
github/kamangir/bluer-uav
ddds
power terminal
D4
SKE6.8A
R1
330 ohm
R2
R3
R15
2.2 kohm
R4
R5
R6
R7
R8
R9
R10
R11
R12
echo1
echo2
echo3
echo4
SW1
blue
green
red
yellow
74.2
49
46
42.3
32.9
31.2
53.3
20.1
8.4
2.4
58
75.8

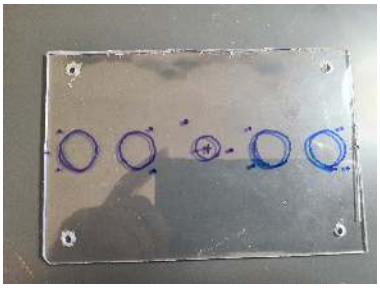
swallow: digital: design: computer: box

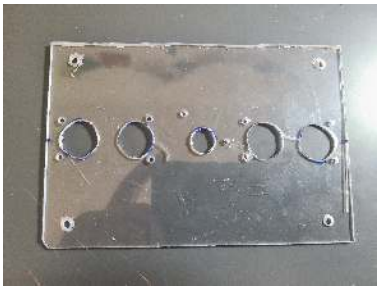








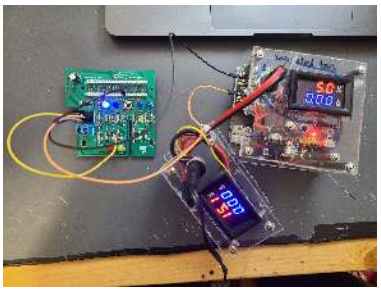


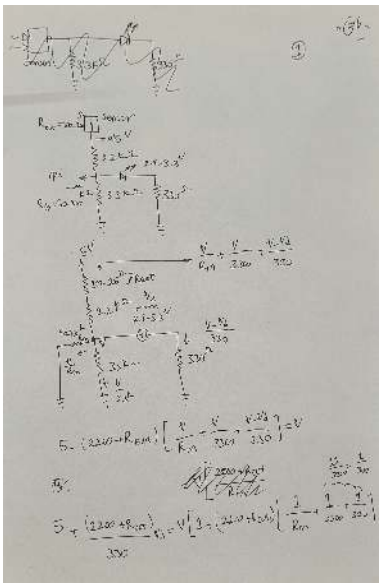


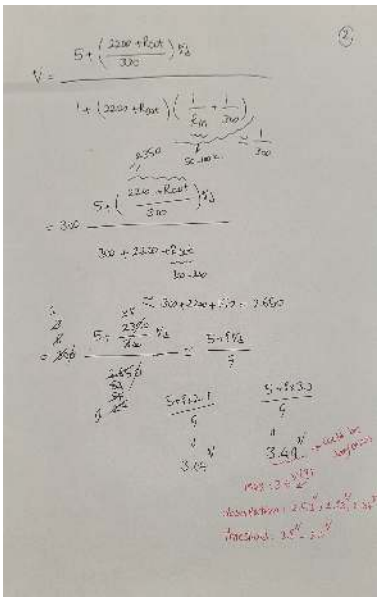
swallow: digital: design: computer: testing

connect adapter-bus and regulated-bus and the additional connections (see figure).

- validate green, red, blue leds.
- validate sw.
- validate echo conditioning. between 2.0 VDC and 3.3 VDC is safe. 2.53 VDC - 2.83 VDC have been observed.

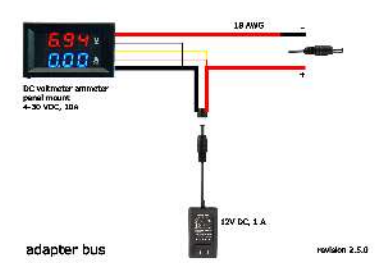


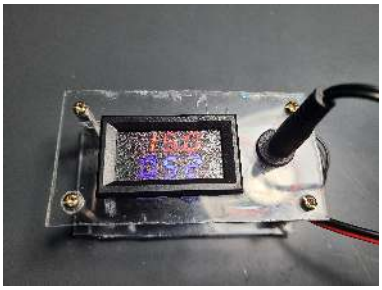


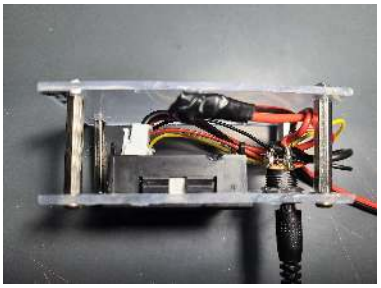
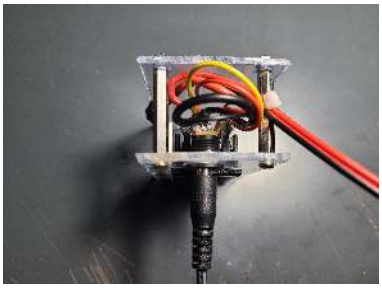
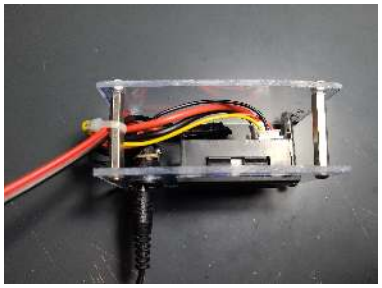
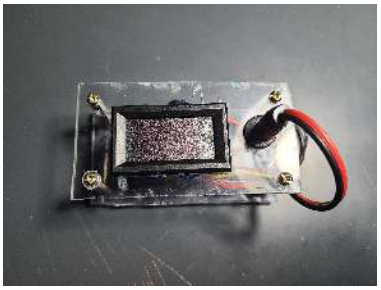
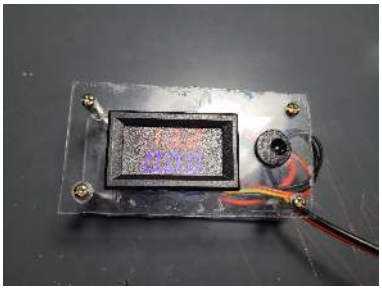


adapter-bus

a non-regulated, ~1 A, DC bus.





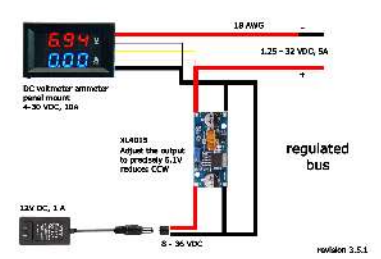

parts

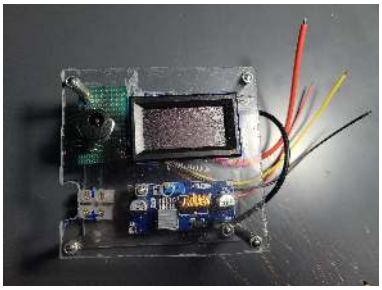
DC power jack, 5.5 mm	DC power plug, 5.5 mm	DSN-VC288, panel mount, 4-30 VDC, 10A, voltmeter ammeter
		

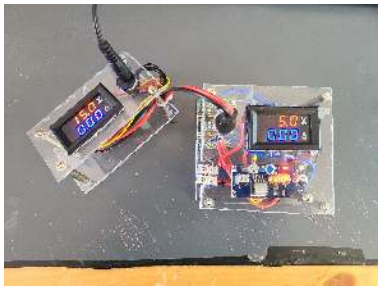
1. DC power jack, 5.5 mm.
2. DC power plug, 5.5 mm.
3. DSN-VC288, panel mount, 4-30 VDC, 10A, voltmeter ammeter.

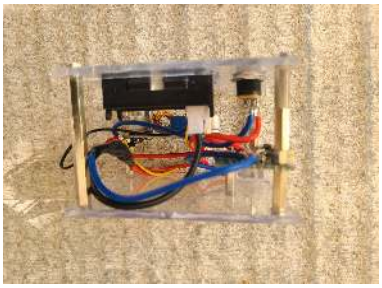
regulated-bus

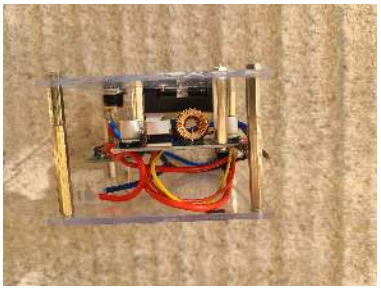
a regulated, 1.25 - 32 VDC, 5A, bus based on XL4015

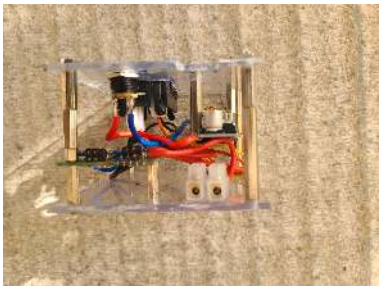


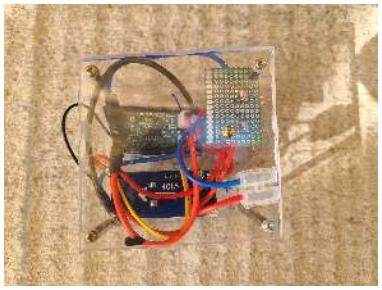


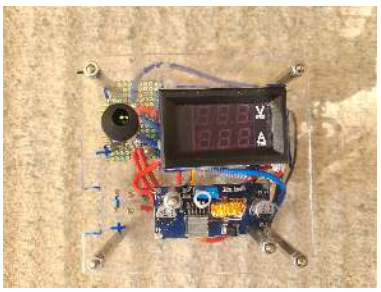


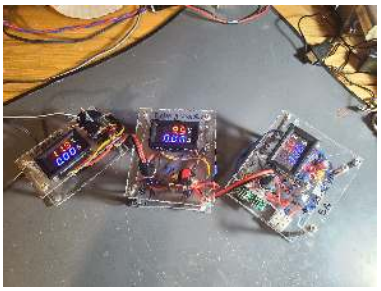












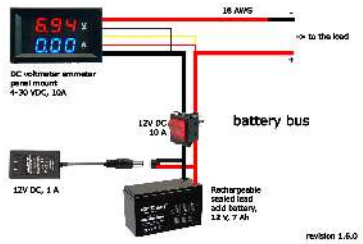
parts

DC power plug, 5.5 mm	DSN-VC288, panel mount, 4-30 VDC, 10A, voltmeter ammeter	XL4015: 8 - 36 VDC -> 1.25 - 32 VDC, 5A	double-sided PCB, 9 cm x 7 cm	nuts, bolts, and spacers	pin headers	plexiglass, 2 mm or 2.5 mm thickness	white terminal
			 10 x 15 holes	 M3: (7 x bolt + 7 x nut + 1 x 5 mm spacer + 7 x 15 mm spacer + 2 x 25 mm spacer + 3 x 35 mm spacer)	 2 x (2 x 3, 90 degree)	2 x 88 cm x 88 cm	 2 x

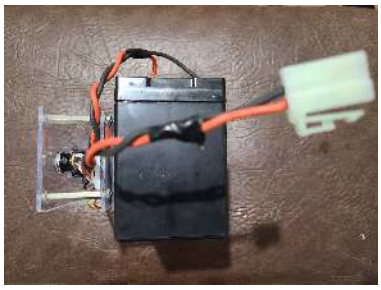
1. DC power plug, 5.5 mm.
2. DSN-VC288, panel mount, 4-30 VDC, 10A, voltmeter ammeter.
3. XL4015: 8 - 36 VDC -> 1.25 - 32 VDC, 5A.
4. double-sided PCB, 9 cm x 7 cm: 10 x 15 holes.
5. nuts, bolts, and spacers: M3: (7 x bolt + 7 x nut + 1 x 5 mm spacer + 7 x 15 mm spacer + 2 x 25 mm spacer + 3 x 35 mm spacer).
6. pin headers: 2 x (2 x 3, 90 degree).
7. plexiglass, 2 mm or 2.5 mm thickness: 2 x 88 cm x 88 cm.
8. white terminal: 2 x.

battery-bus

a non-regulated ~12 VDC, ~3 A, DC bus.



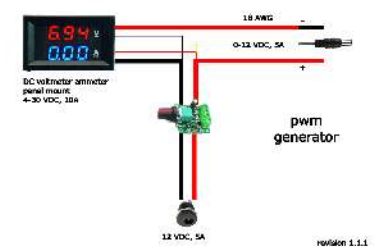



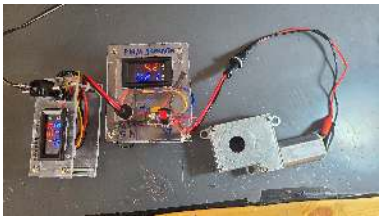
parts

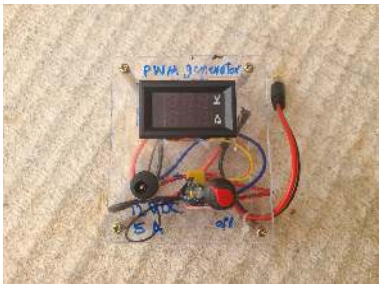
DC power plug, 5.5 mm 	DSN-VC288, panel mount, 4-30 VDC, 10A, voltmeter ammeter 	Rechargeable sealed lead acid battery  12 V, 7.2 Ah, or more.	on/off DC switch with indicator led  12V DC 10 A

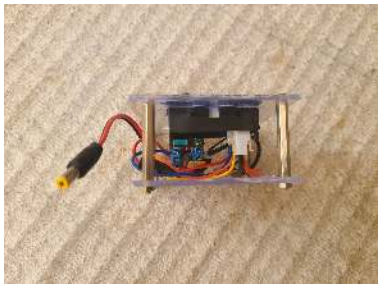
1. DC power plug, 5.5 mm.
2. DSN-VC288, panel mount, 4-30 VDC, 10A, voltmeter ammeter.
3. Rechargeable sealed lead acid battery: 12 V, 7.2 Ah, or more..
4. on/off DC switch with indicator led: 12V DC 10 A.

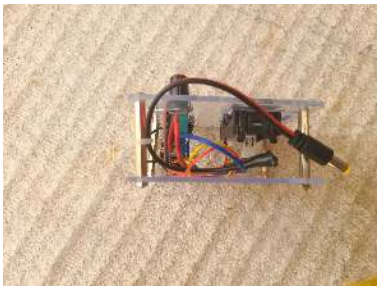
pwm-generator

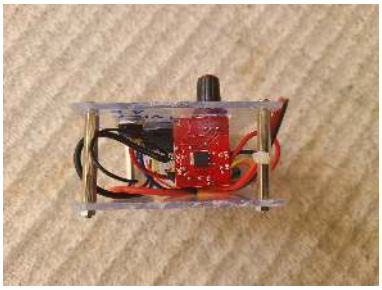


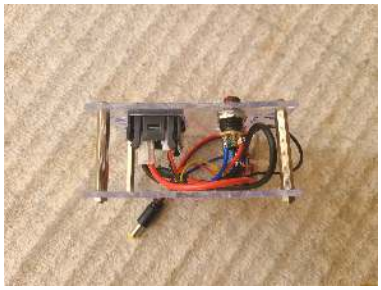


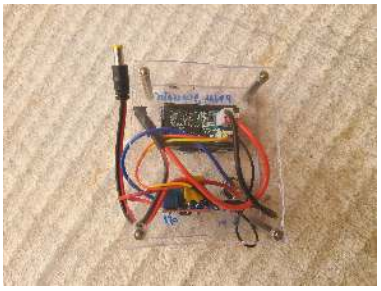












parts

DC power jack, 5.5 mm 	DC power plug, 5.5 mm 	DSN-VC288, panel mount, 4-30 VDC, 10A, voltmeter	nuts, bolts, and spacers 	plexiglass, 2 mm or 2.5 mm thickness	pwm manual DC motor controller,

		ammeter 	M3: (4 x bolt + 4 x nut + 4 x 40 mm spacer)	 2 x 80 cm x 100 cm	12 V, ≥ 5 A 

1. DC power jack, 5.5 mm.
2. DC power plug, 5.5 mm.
3. DSN-VC288, panel mount, 4-30 VDC, 10A, voltmeter ammeter.
4. nuts, bolts, and spacers: M3: (4 x bolt + 4 x nut + 4 x 40 mm spacer).
5. plexiglass, 2 mm or 2.5 mm thickness: 2 x 80 cm x 100 cm.
6. pwm manual DC motor controller, 12 V, ≥ 5 A.

swallow: digital: design: computer: power

swallow head: - Raspberry Pi 4B (CPU/GPU busy, Wi-Fi + BLE on): ~ 1.3–1.5 A -> ~ 6.6–7.7 W - Raspberry Pi Camera (capturing video): ~ 200–250 mA -> ~ 1.0–1.3 W - 4 × HC-SR04 (all active): ~ 15 mA each → ~ 60 mA -> ~ 0.3 W - HDMI connection: ~ 50 mA -> 0.25 W

total: ~ 10 W ~= ~1.8 A @ 5.1 V DC ~= 1 A @ 12 V DC (90% efficiency)

swallow: - 4 x BTS7960 logic side: ~ 10–20 mA -> 0.1 W (will consider included in ⬆️) - 4 x DC motors: 1 A (freely-rotating) - 4 A (stalled) per motor ==

4A ~= 8 A -> ~ 100 W

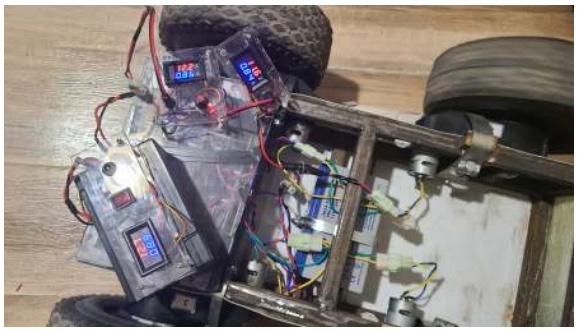
total: - mostly idling: 1 A == 12 W - cruising driving: 1 A + 4 x 1 A = 5 A == 60 W - aggressive driving: 1 A + 4 x 2 A = 9 A == 108 W

option	battery	rated Wh	usable Wh	aggressive Driving	cruising driving	mostly idling
SLA						
1	7.2 Ah ⭐	86 Wh	50–65 Wh	28–37 min	50–70 min	4–5.5 h
2	12 Ah	144 Wh	85–110 Wh	48–62 min	1.4–1.8 h	7–9 h
3	18 Ah	216 Wh	130–165 Wh	73–92 min	2.2–2.8 h	11–14 h
4	20 Ah	240 Wh	145–180 Wh	81–100 min	2.5–3.0 h	12–15 h
5	30 Ah	360 Wh	215–270 Wh	2.0–2.5 h	3.6–4.6 h	18–22 h
LiPo						
6	7 Ah	78 Wh	66 Wh	37 min	67 min	33.0 h ❓
7	10 Ah	111 Wh	94 Wh	52 min	96 min	47.2 h ❓
8	15 Ah	166 Wh	142 Wh	79 min	2.40 h	70.8 h ❓
9	20 Ah	222 Wh	189 Wh	105 min	3.20 h	94.3 h ❓
10	25 Ah	278 Wh	236 Wh	2.20 h	4.00 h	117.9 h ❓

option	battery	rated Wh	usable Wh	aggressive Driving	cruising driving	mostly idling
11	30 Ah	333 Wh	283 Wh	2.65 h	4.80 h	141.5 h ?

SLA: 12 V, 60–75% usable capacity.

LiPo: 3S, 11.1 V nominal, 85% usable capacity.



SLA vs. LiPo

- Weight: LiPO ≈ ⅓ the weight of SLA for the same Ah.
- Cost in Iran: LiPO ≈ 2–3× more expensive than SLA per Ah.
- Lifetime: LiPO lasts 5–10× more cycles (much cheaper per cycle).
- Safety:
  - SLA: very stable, heavy, low fire risk.
  - LiPO: safe if BMS is good; RC LiPo is riskier.
- Complexity:
  - SLA: simple, just charge and use.
  - LiPO: needs proper charger + BMS awareness, but gives better performance.

swallow: digital: design: computer: naming

- `<ugv-name>` or `<ugv-name>-front`: a swallow computer that watches the front and controls the motors.
- `<ugv-name>-back`: a swallow-head computer that watches the back.
- 2 x `<ugv-name>-top`: two swallow-head computer that watch the sides (`-left` and `-right`).

swallow: digital: design: rpi-pinout

Responsibility	Function	Physical Pin	GPIO	Notes
Motor 1, FW	PWM	12	18	PWM0, Steering / Right
Motor 1, BW	PWM	32	12	PWM0 (alternate), Shares PWM0 with GPIO 18
Motor 2, FW	PWM	33	13	PWM1, Rear / Left
Motor 2, BW	PWM	35	19	PWM1 (alternate), Shares PWM1 with GPIO 13
Green LED	Digital Output	11	17	
RED LED	Digital Output	13	27	
Blue LED	Digital Output	15	22	
Push Button	Digital Input	37	26	
Sensor 1, Trigger	Digital Output	16	23	
Sensor 2, Trigger	Digital Output	29	5	
Sensor 3, Trigger	Digital Output	31	6	
Sensor 4, Trigger	Digital Output	36	16	
Sensor 1, Echo	Digital Input	18	24	
Sensor 2, Echo	Digital Input	22	25	
Sensor 3, Echo	Digital Input	38	20	
Sensor 4, Echo	Digital Input	40	21	

swallow: digital: design: operation

keyboard

event	full keyboard	numpad
debug off	v	9
debug on	b	7
exit	*i	*7
mode = action	g	1
mode = none	y	5
mode = training	t	3
reboot	*p	*5
shutdown	*o	*9
special key	z	.
speed backward	s	2
speed forward	w	8
steer left	a	4
steer right	d	6
stop		0
ultrasonic off	n	-
ultrasonic on	m	+
update	*u	*1

*: special key.

to enable full keyboard:

```
@swallow env set full_keyboard 1
```

the range of numpad is ~10-20 m range, noticeably lower than that of the full keyboard, which is ~50 m, see village-6 for details.

leds

- green: control loop.
- red:
    - flashing:
        - motor update.
        - action / training.
        - release the push button to update.
        - setpoint update.
    - solid: release the push button to shutdown.
- yellow:
    - command received.
    - mousepad activity.
- blue: ultrasonic sensor echo.

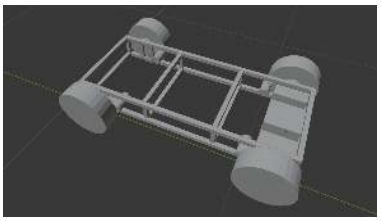
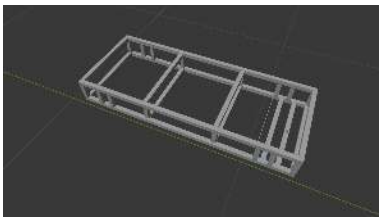
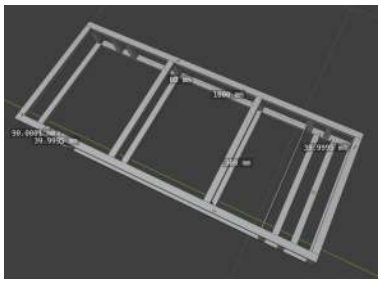
▶ mouse pad (obsolete)

push button

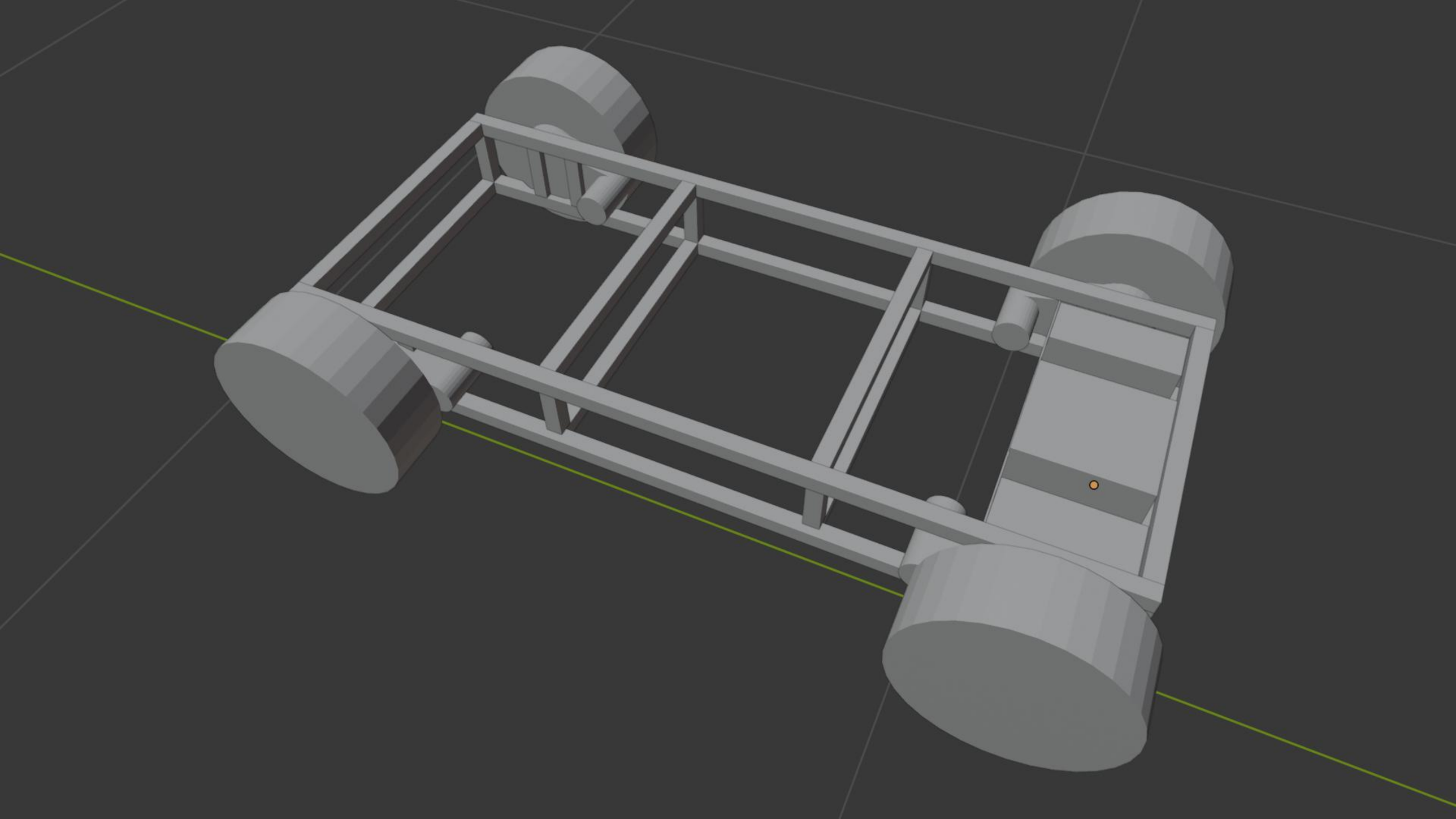
- hold for 5 seconds: update.
- hold for 10 seconds: shutdown.
- hold for > 15 seconds: skip.

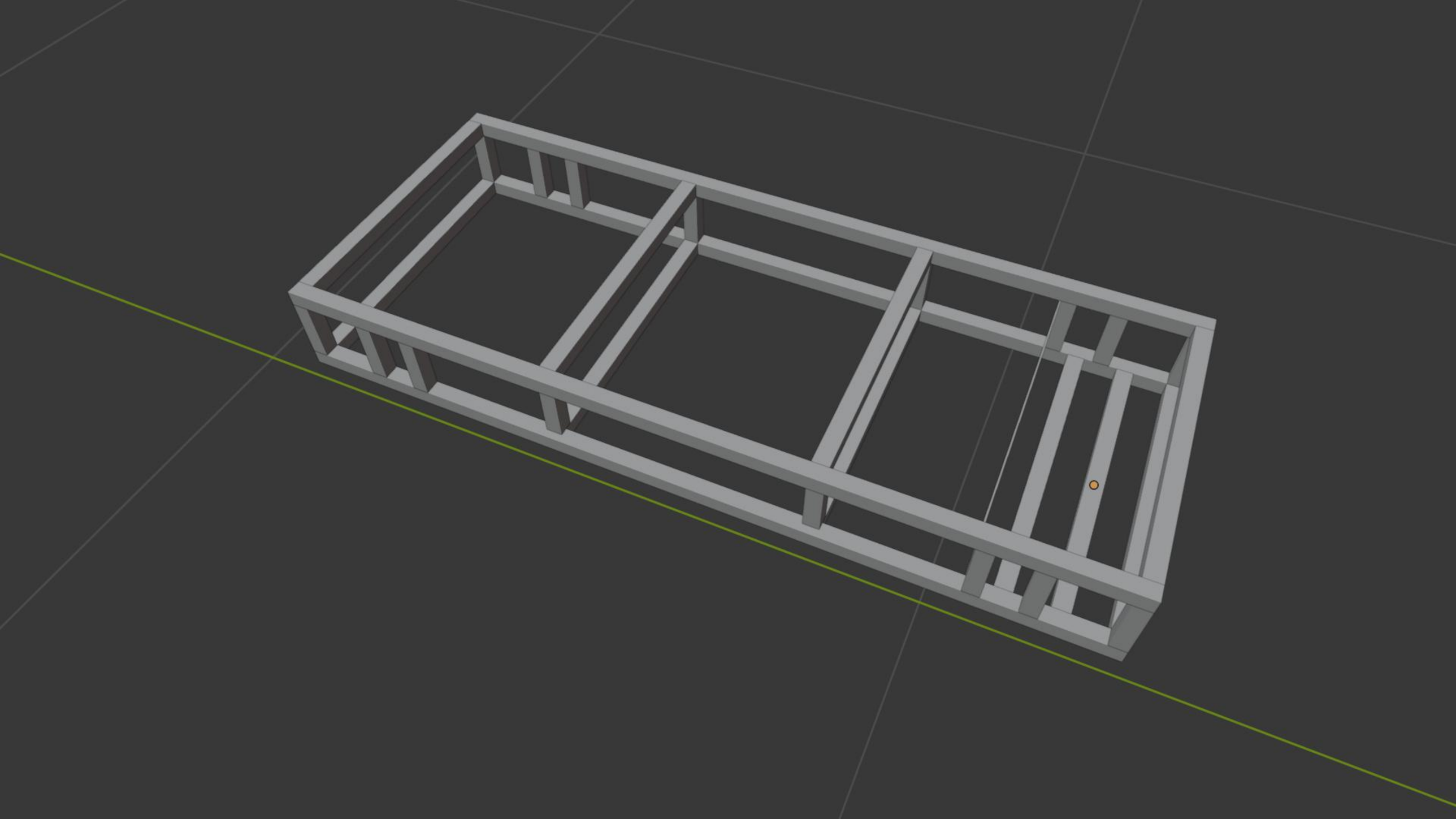
swallow: digital: design: mechanical

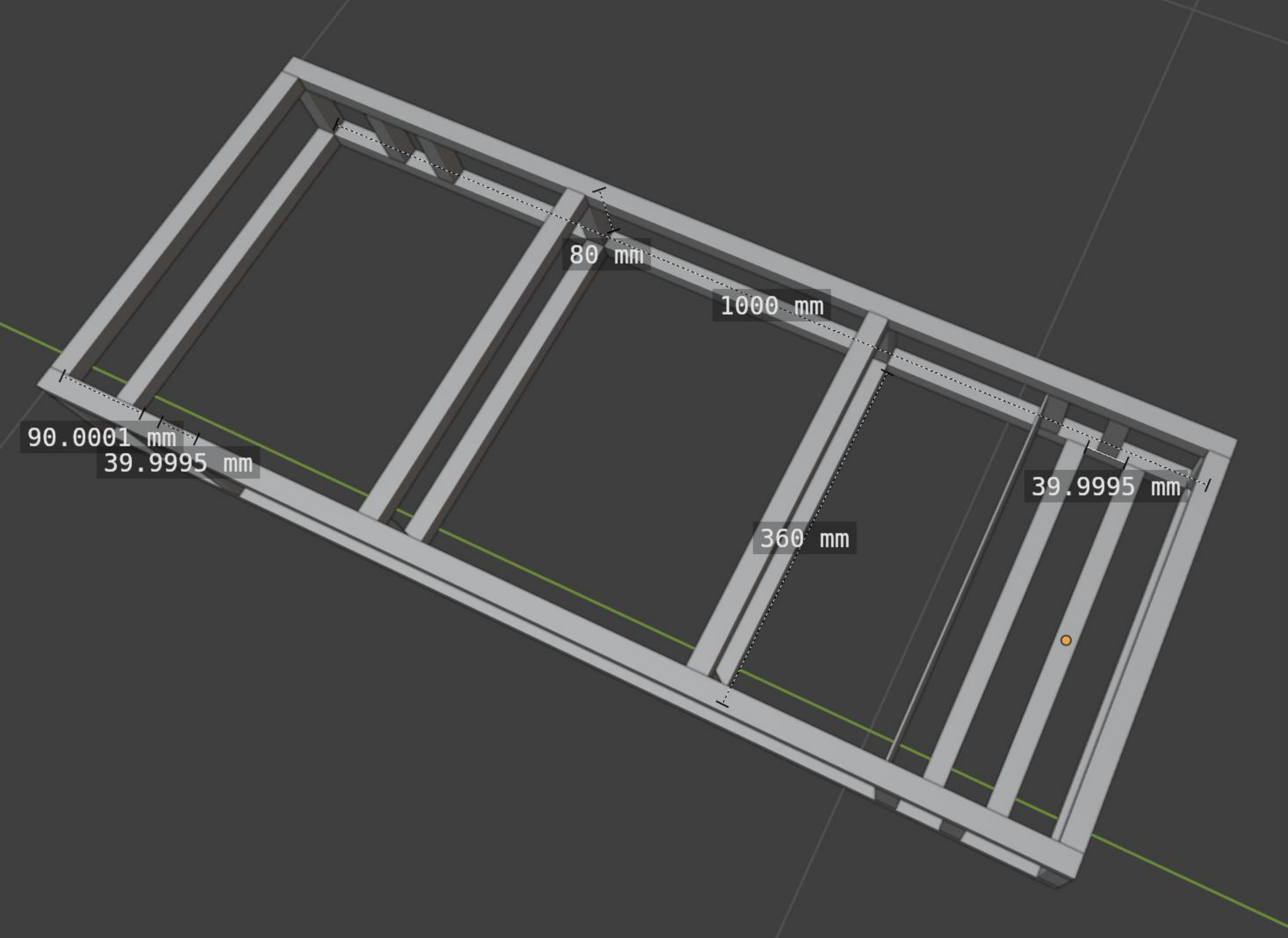
- Blender files
- material: 2 x 20 mm x 20 mm x 1.5 mm x 6000 mm
- parts:
  - 4 x 1000 mm
  - 10 x 360 mm
  - 16 x 80 mm
  - total: 8520 mm (8.5 m)
  - cut into 4 x (1000 mm + 360 mm = 1360 mm) + 3 x (360 mm + 4 x 80 mm = 680 mm) + (3 x 360 mm + 4 x 80 mm = 1400 mm) ~= 1500 mm

- v1






80 mm
1000 mm
90.0001 mm
39.9995 mm
360 mm
39.9995 mm

swallow: digital: design: ultrasonic-sensor: dev

test

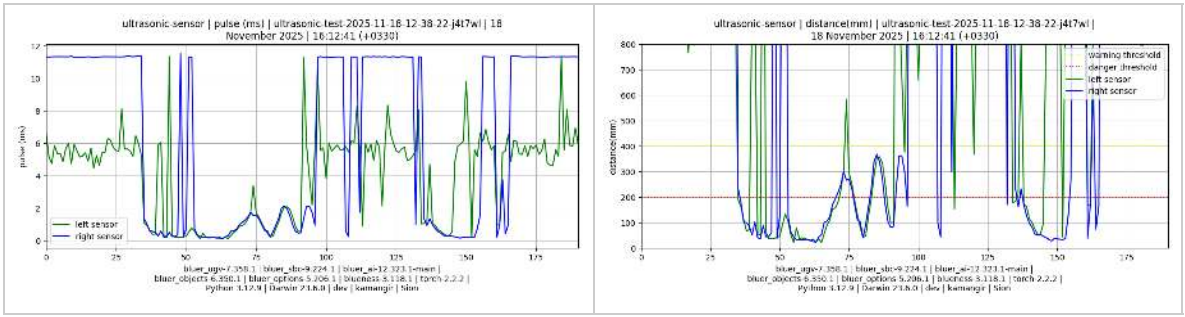
using ultrasonic_sensor-v8.py.

```
🐬 left: no detection  | pulse= 11.33 ms | dist≈ 1943 mm
🐬 left: no detection  | pulse= 11.37 ms | dist≈ 1949 mm
🐬 left: detection  | pulse=  0.93 ms | dist≈  159 mm
🐬 left: detection  | pulse=  0.98 ms | dist≈  168 mm
```

using ultrasonic_sensor-v9.py.

```
🐬 left    : no detection    ,  11.31 ms ==  1939 mm | right   : no detection    ,  11.32 ms ==  1941 mm
🐬 left    : no detection    ,   5.27 ms ==   904 mm | right   : no detection    ,  11.28 ms ==  1935 mm
🐬 left    : detection       ,   4.50 ms ==   772 mm | right   : no detection    ,   4.75 ms ==   814 mm
🐬 left    : detection       ,   4.34 ms ==   745 mm | right   : detection ,   4.63 ms ==   795 mm
```

```
@rpi
@select ultrasonic-test-$(@timestamp)

@swallow ultrasonic test - .
@.

@mac
@select $BLUER_UGV_ULTRASONIC_SENSOR_TEST_OBJECT

@assets publish \
    extensions=png,push

@upload public,zip
@.
```



ultrasonic-test-2025-11-18-12-38-22-j4t7wl

review

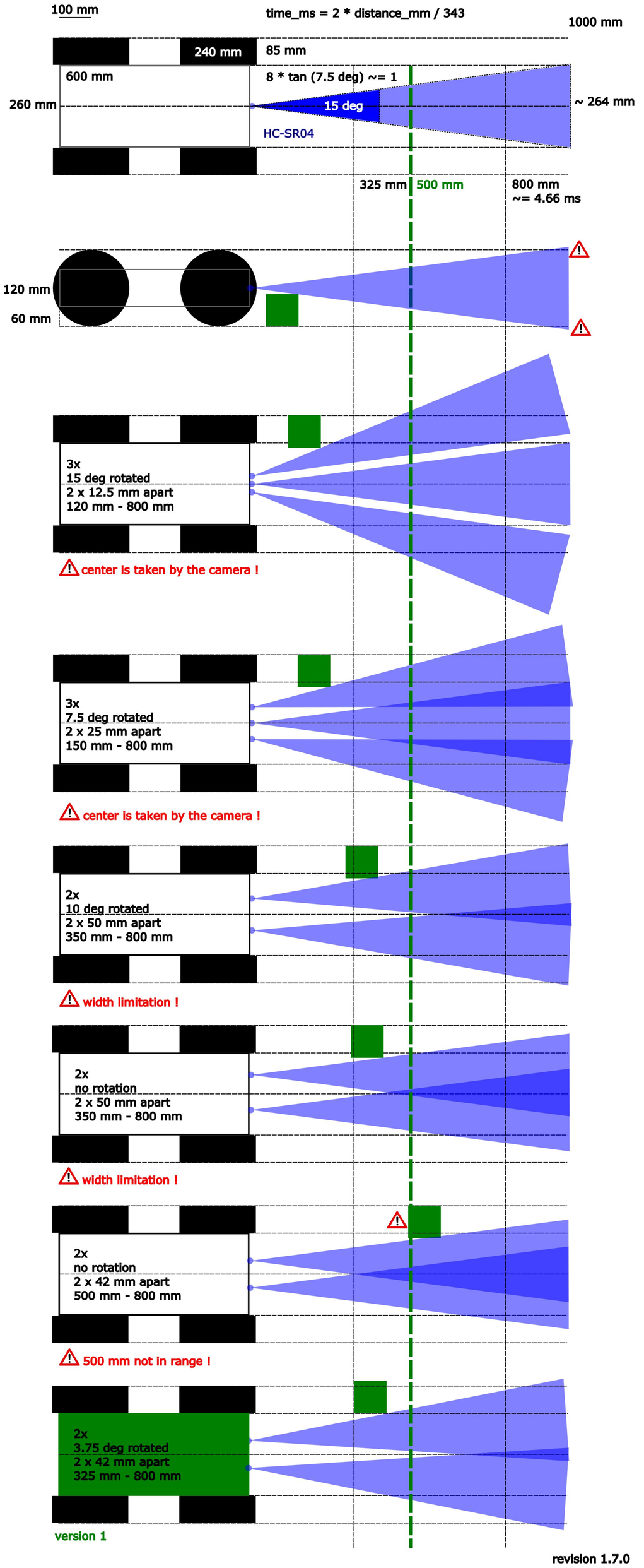
```
@select $BLUER_UGV_ULTRASONIC_SENSOR_TEST_OBJECT

@swallow ultrasonic review download

@assets publish \
    extensions=gif,push

@upload public,zip
```

image
image

ultrasonic-test-2025-11-18-12-38-22-j4t7wl

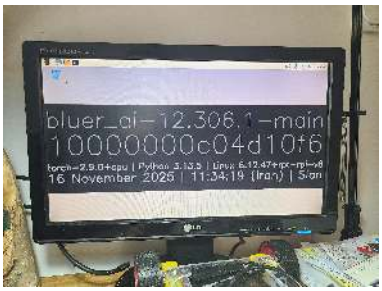
in session

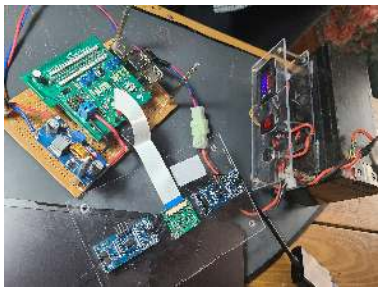
ultrasonic sensors: geometry design

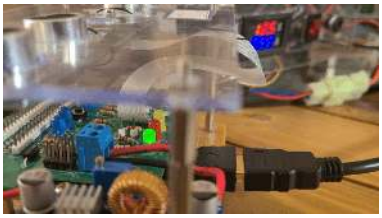


swallow: digital: design: testing

- test the computer.
- disconnect the shield from the XL4015 and connect the computer to the battery bus. validate that the red LED on the XL4015 turns on.
- adjust XL4015 at 5.1 V.
- separate the shield from the rpi, connect power to the shield, turn the power on, validate no 💥, turn the power off.
- install the shield on the rpi, screw the shield, connect the monitor (for rpi4b: to the hdmi port closer to the USB port) and the keyboard, turn the power on, validate full operation, shutdown, power off.
- wire the ultrasonic sensors, turn the power on, validate that ultrasonic sensors log warning and danger, `@swallow debug`, test the camera, shutdown, power off.





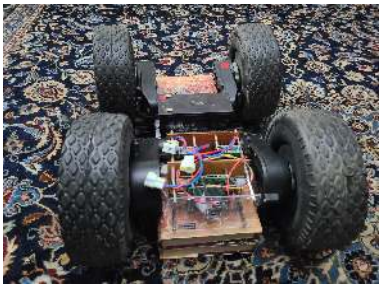
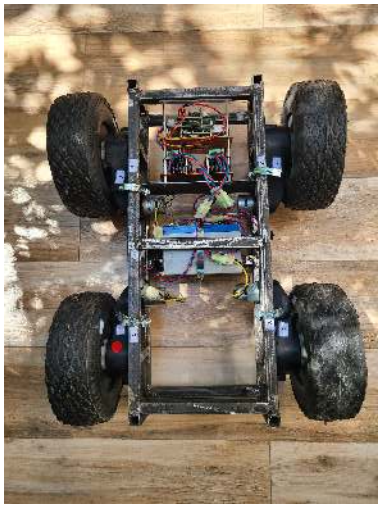
arzhang

> swallow's little sister, formerly known as `sparrow`, `arzhang` (ارژنگ) is named after the demon in the Shahnameh. It embodies noble strength and quiet defense.

- design
- algo






















arzhang: design: specs

- dimensions: 300 mm x 600 mm
- speed: 10 km / h (max)
- weight: ~15 kg
- wheel diameter: 20 cm (270 RPM) - 30 cm (180 RPM)
- motor + gearbox: metal, 90-degree, 180 - 270 RPM, 12 V, 10 kg cm torque.
- drives: 50 – 100 W

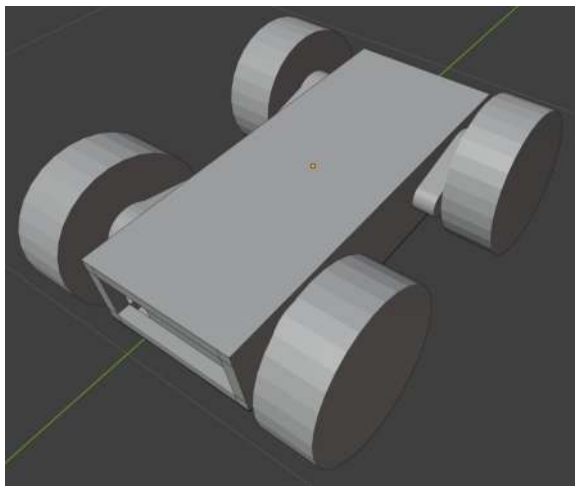
speed calculations

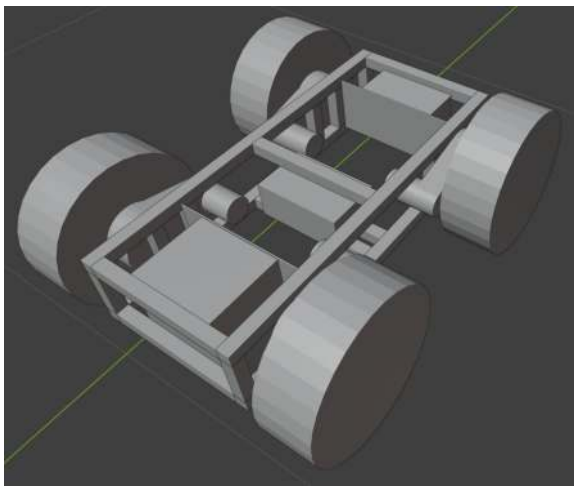
- 90 RPM * 20 cm * 3.14 = 5,652 cm / min = 56.52 m / min == 3,4 km / h
- 270 RPM ~= 10 km / h

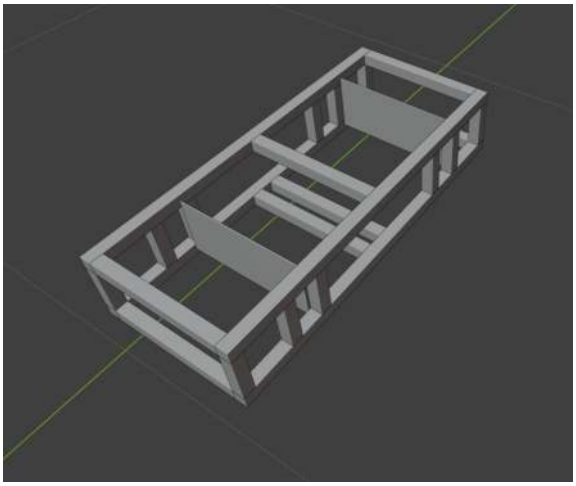
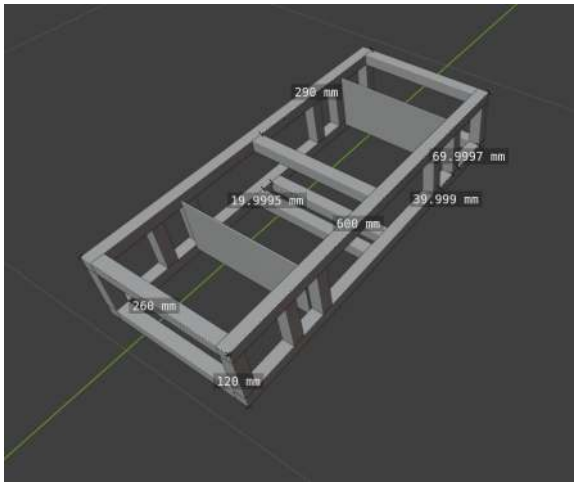
arzhang: design: mechanical

- Blender files
- metal profile: 20 mm x 20 mm x 1.5 mm x 6000 mm
  - parts:
    - 4 x 600 mm
    - 7 x 220 mm
    - 12 x 80 mm
    - total: 5060 mm (5.06 m)
  - cut into 4 x (600 mm + 2 x 220 mm + 3 x 80 mm = 1280 mm) ~= 1500 mm
- metal sheet:
  - 2 x 220 mm x 120 mm x 2 mm (1, if one-headed)
- plexiglass
  - 260 mm x 600 mm x 2 mm
- thin metal sheet
  - 260 mm x 600 mm x 0.5 mm
- fiberglass
  - 2 x 120 mm x 600 mm
  - 260 mm x 120 mm (if one-headed)
- velcro
  - 1720 mm x 20 mm

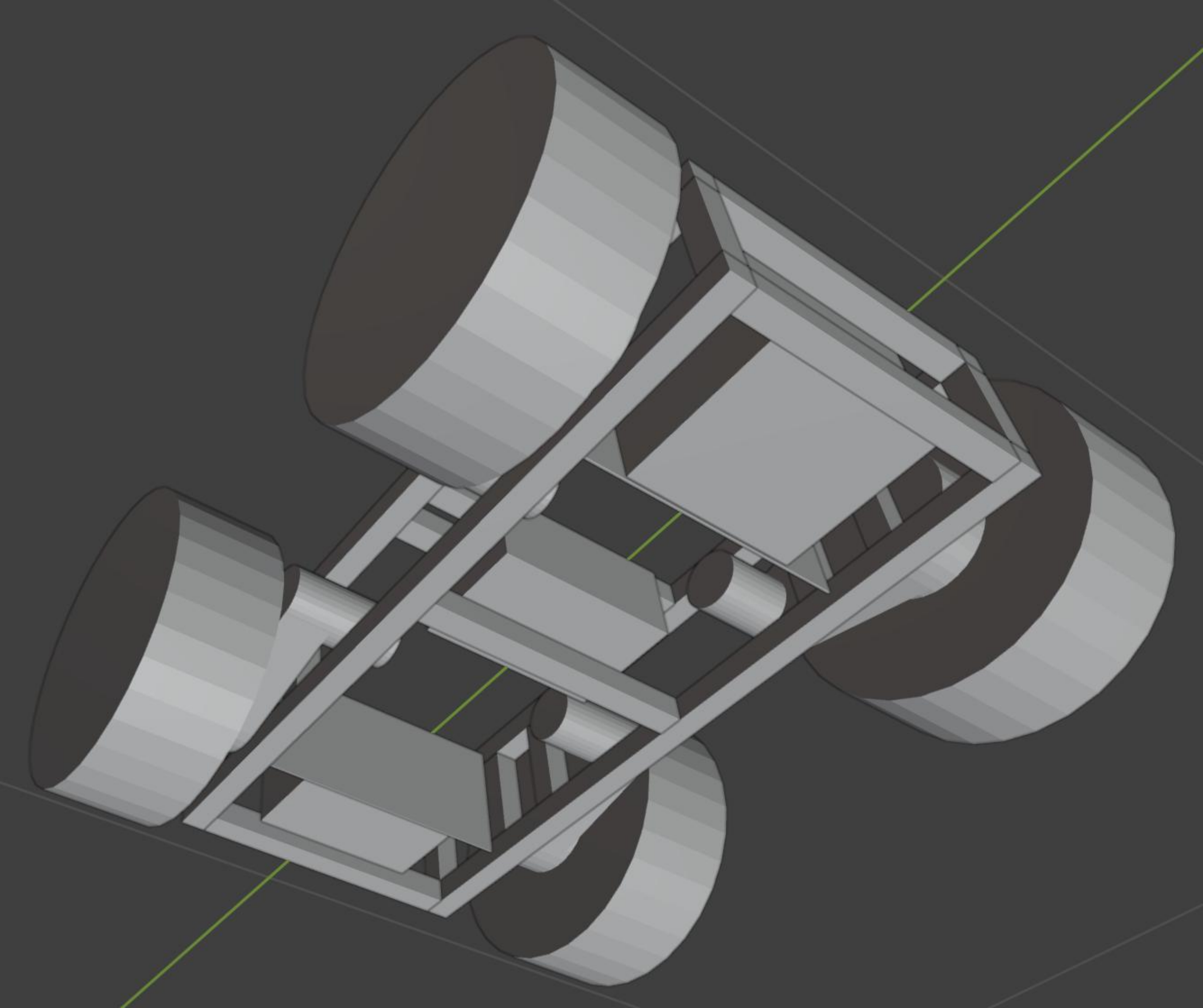
⏰ mechanical build takes ~4 hours.

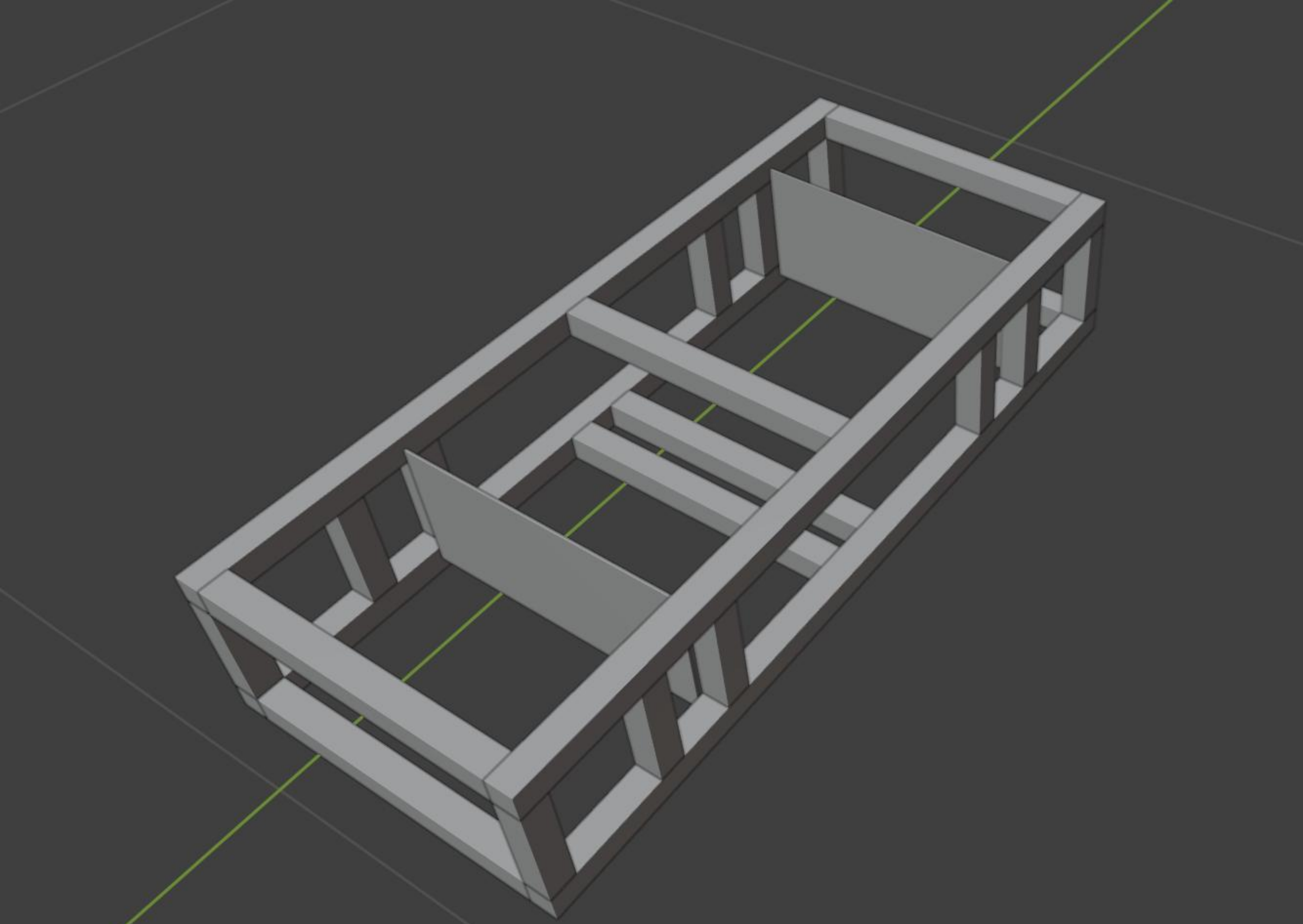


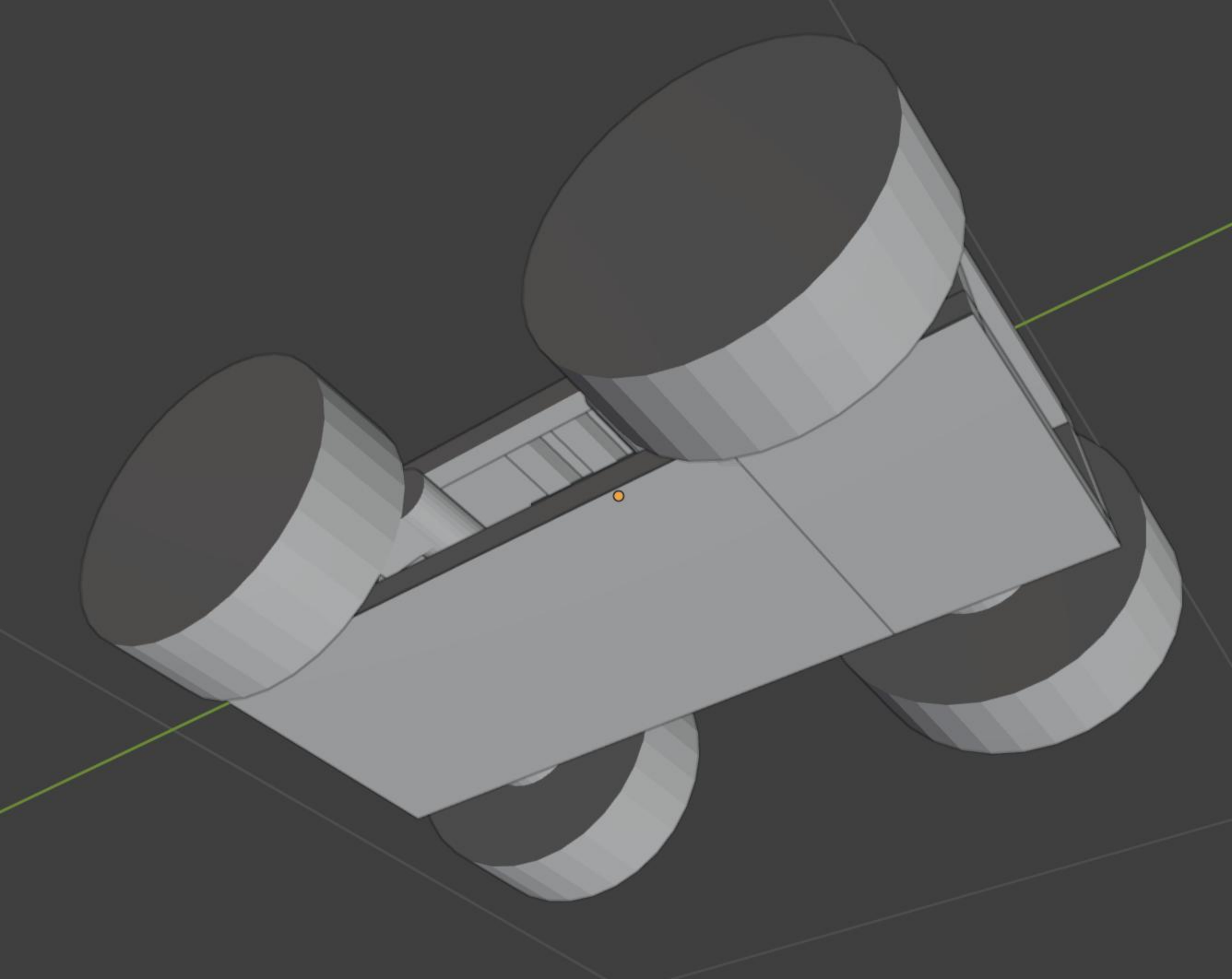


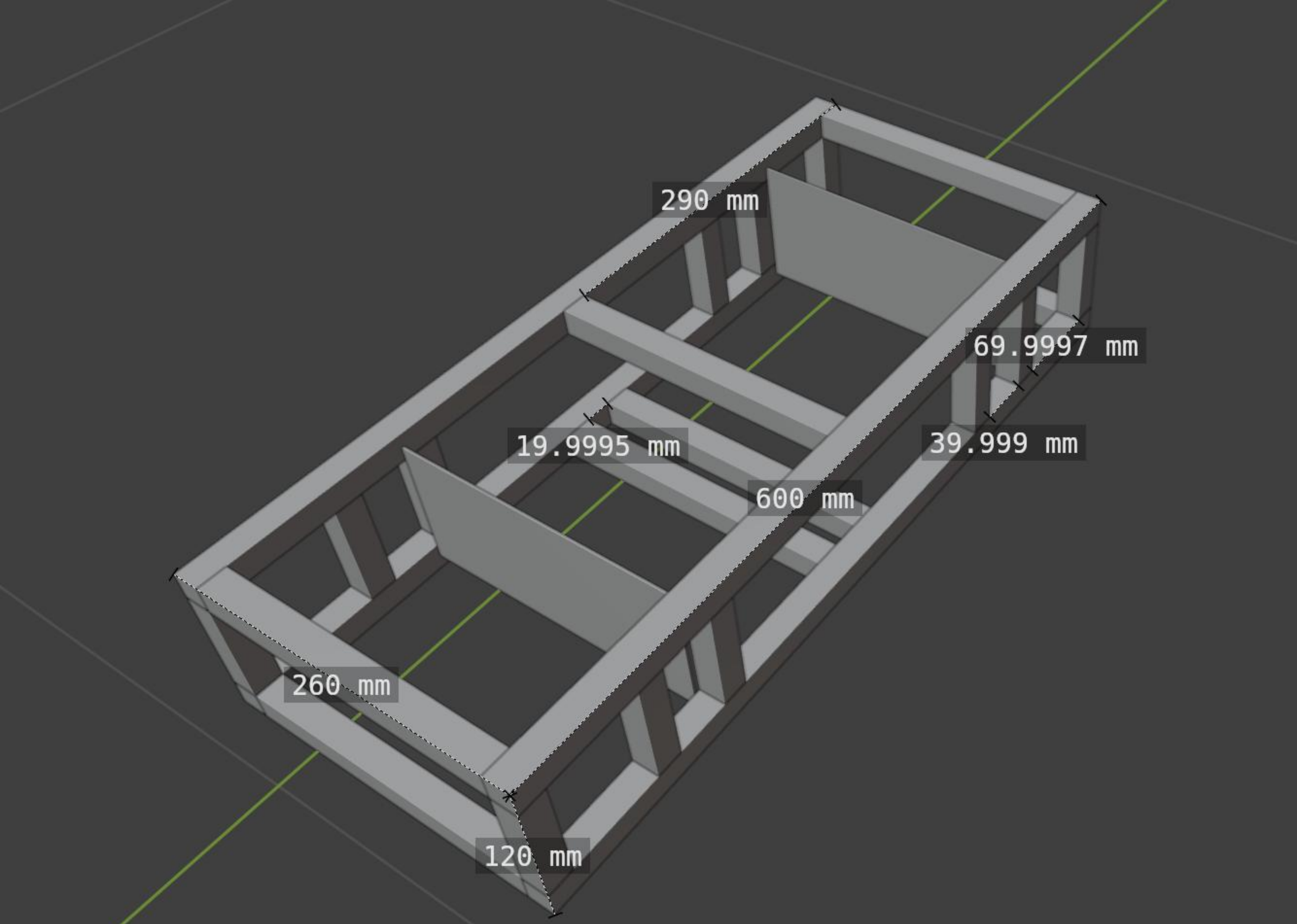
<table>
<tr>
<td>  </td>
<td> 
</td>
</tr>
<tr>
<td>  </td>
<td>  </td>
</tr>
</table>

- v1








290 mm
69.9997 mm
19.9995 mm
39.999 mm
600 mm
260 mm
120 mm

arzhang: design: power

- uses swallow computer power.
  - swallow: 60 W
  - swallow-head: 10 W
- total: 70 W ~= 6 A @ 12 V DC

runtime	energy needed	ideal capacity @ 12 V	LiPo Ah (practical)	SLA Ah (practical)
1 h	70 W	6 Ah	7 Ah	12 Ah ⭐
2 h	140 W	12 Ah	13 Ah	24 Ah
3 h	210 W	18 Ah	19 Ah	36 Ah
4 h	280 W	23 Ah	26 Ah	48 Ah
5 h	350 W	29 Ah	32 Ah	60 Ah

SLA: 85%, LiPo: 90%

swallow: digital: algo: driving

driving with a keyboard.

```
@swallow env cp driving
```

swallow: digital: algo: navigation

navigation using an @algo/image_classifier.

```
@swallow env cp navigation
```

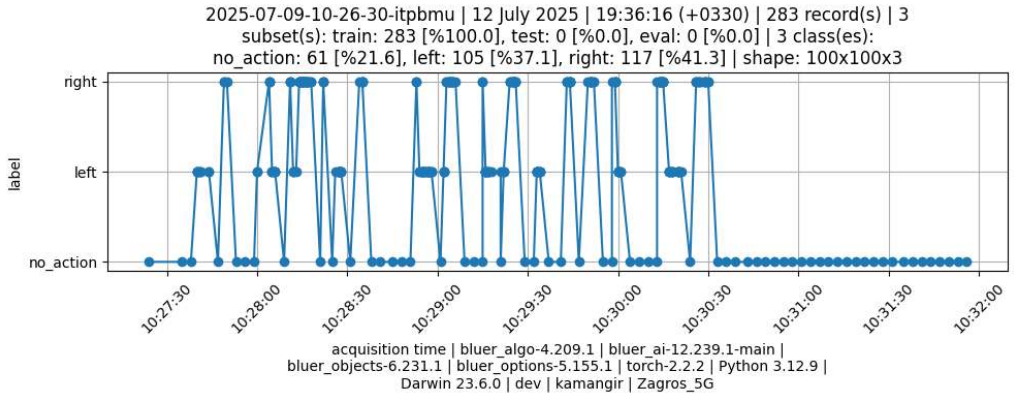
- dataset
- model

swallow: digital: algo: navigation: dataset: collection: validation

Start `swallow`, press `t` (to start training), drive for 5 minutes, press `i` (to exit).

```
@select
@session start

@select 2025-07-09-10-26-30-itpbmu

@download - .
@upload public,zip .
@assets publish \
    extensions=png,push . \
    --prefix grid
```

image



image

[2025-07-09-10-26-30-itpbmu]

```
dataset:
  class_count: 3
  classes:
    0: no_action
    1: left
    2: right
  count: 283
  shape:
  - 100
  - 100
  - 3
  source: 00000000c74cf7d2
  subsets:
    eval: 0
    test: 0
    train: 283
```

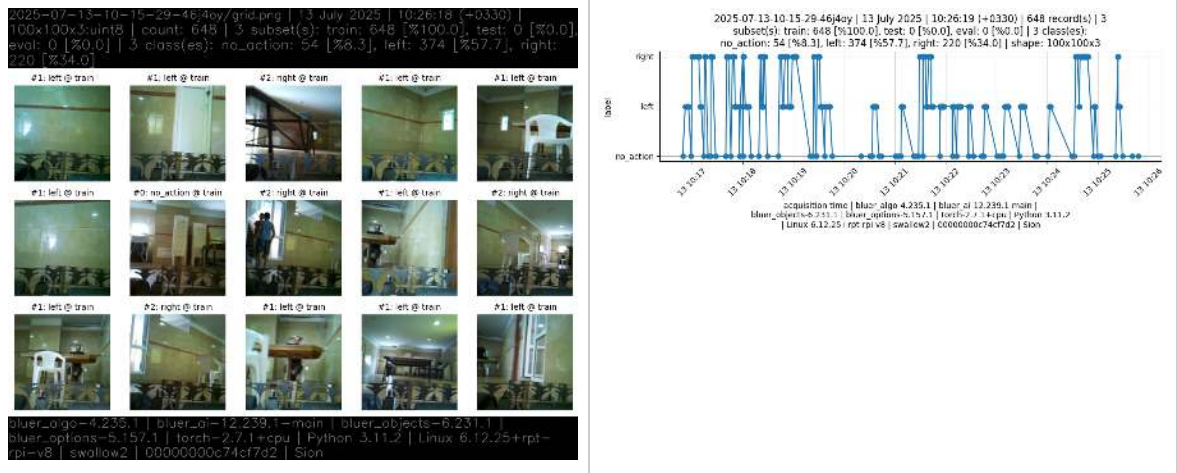
swallow: digital: algo: navigation: dataset: collection: one

```
@list log \
    $(@list filter \
        $(@swallow dataset list) \
        --contains $(@today))
```

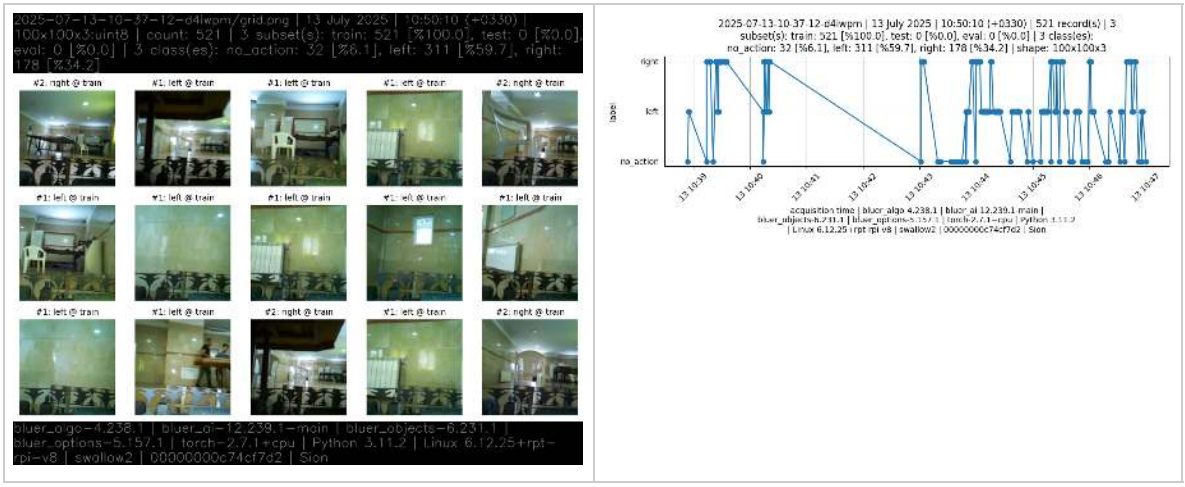
list of 3 item(s): 2025-07-13-10-15-29-46j4oy, 2025-07-13-10-37-12-d4iwpm, 2025-07-13-12-55-54-cx5mhk.

```
runme() {
    local object_name
    for object_name in $(@list filter \
        $(@swallow dataset list) \
        --contains $(@today) | tr , " "); do
        @select $object_name

        @download policy=doesnt_exist .
        @upload public,zip .
        @assets publish \
            extensions=png,push . \
            --prefix grid
    done
}

runme
```

2025-07-13-10-15-29-46j4oy



2025-07-13-10-37-12-d4iwpm

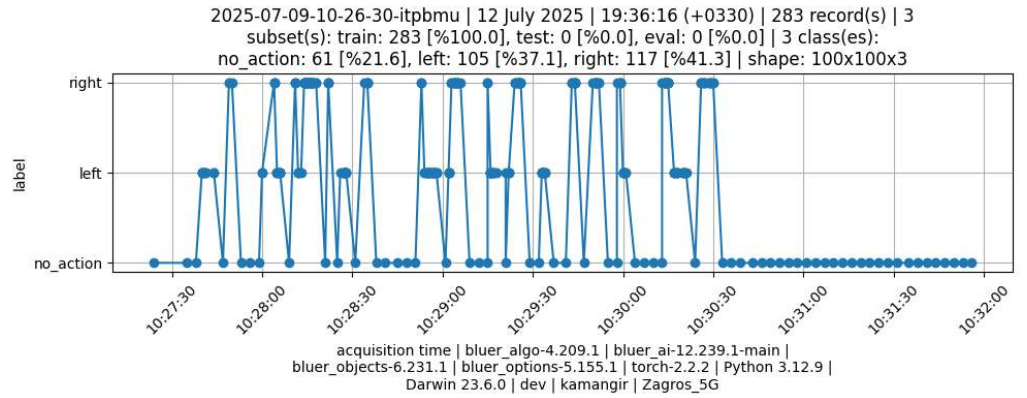


2025-07-13-12-55-54-cx5mhk

swallow: digital: algo: navigation: dataset: review

```
@select 2025-07-09-10-26-30-itpbmu

@algo image_classifier dataset review - .

@upload public,zip .
@assets publish \
    extensions=png,push . \
    --prefix grid
```

image



image

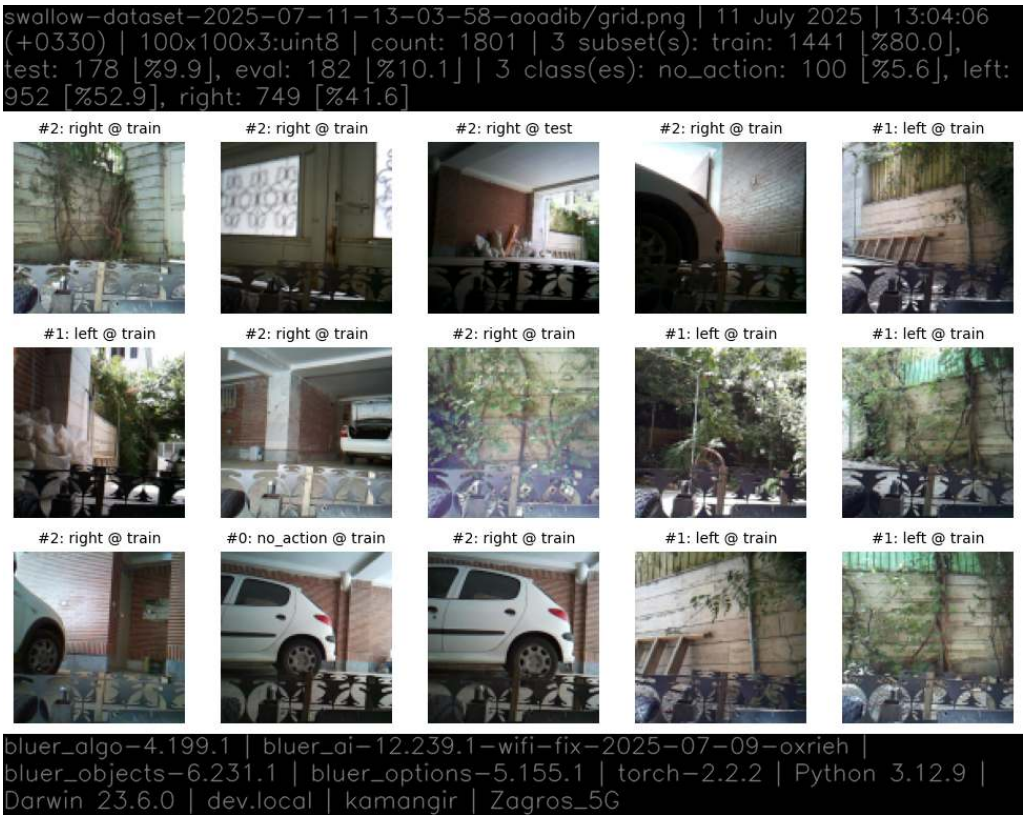
2025-07-09-10-26-30-itpbmu

```
dataset:
  class_count: 3
  classes:
    0: no_action
    1: left
    2: right
  count: 283
  shape:
  - 100
  - 100
  - 3
  source: 00000000c74cf7d2
  subsets:
    eval: 0
    test: 0
    train: 283
```

swallow: digital: algo: navigation: dataset: combination: validation

uses collection/validation.

```
@select swallow-dataset-$(@timestamp)

@swallow dataset combine \
    count=2 .

@upload public,zip .
@assets publish \
    extensions=png,push . \
    --prefix grid
```



image

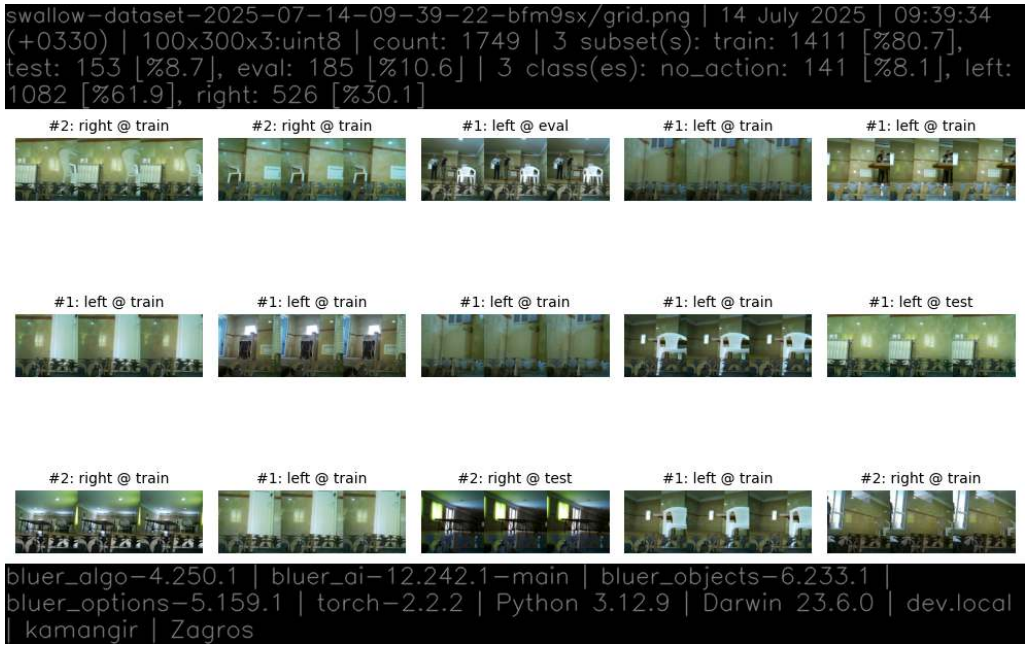
[swallow-dataset-2025-07-11-13-03-58-aoadib]

```
dataset:
  class_count: 3
  classes:
    0: no_action
    1: left
    2: right
  contains:
  - 2025-07-09-11-16-52-4zo4zc
  - 2025-07-09-11-34-19-bcoh75
  count: 1801
  shape:
  - 100
  - 100
  - 3
  subsets:
    eval: 182
    test: 178
    train: 1441
```

swallow: digital: algo: navigation: dataset: combination: one

uses collection/one.

```
@select swallow-dataset-$(@timestamp)

@swallow dataset combine \
    sequence=3 . \
    --datasets $(@list filter \
        $(@swallow dataset list) \
        --contains 2025-07-13)

@upload public,zip .
@assets publish \
    extensions=png,push . \
    --prefix grid
```



image

swallow-dataset-2025-07-14-09-39-22-bfm9sx

```
dataset:
  class_count: 3
  classes:
    0: no_action
    1: left
    2: right
  contains:
  - 2025-07-13-10-15-29-46j4oy
  - 2025-07-13-10-37-12-d4iwpm
  - 2025-07-13-12-55-54-cx5mhk
  count: 1749
  shape:
```

```
- 100
- 300
- 3
subsets:
  eval: 185
  test: 153
  train: 1411
```

swallow: digital: algo: navigation: model: validation

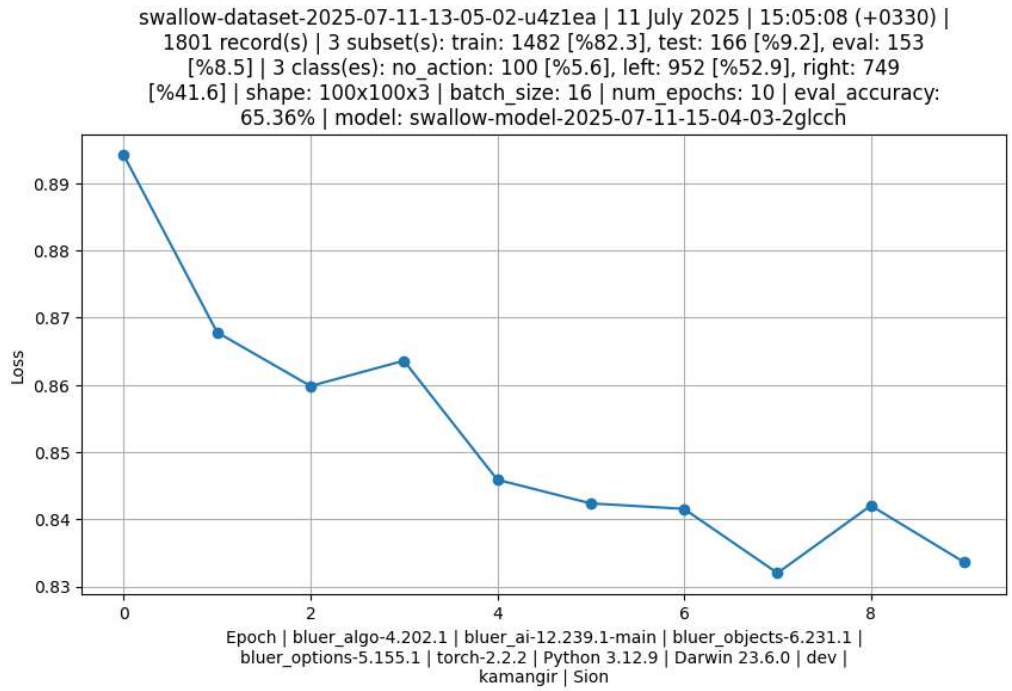
uses combination/validation.

```
@select swallow-dataset-$(@timestamp)

@swallow dataset combine \
    count=2 .

@upload public,zip .
@assets publish \
    extensions=png,push . \
    --prefix grid

@select swallow-model-$(@timestamp)

@image_classifier model train upload .. .

@upload public,zip .
@assets publish \
    extensions=png,push .
```

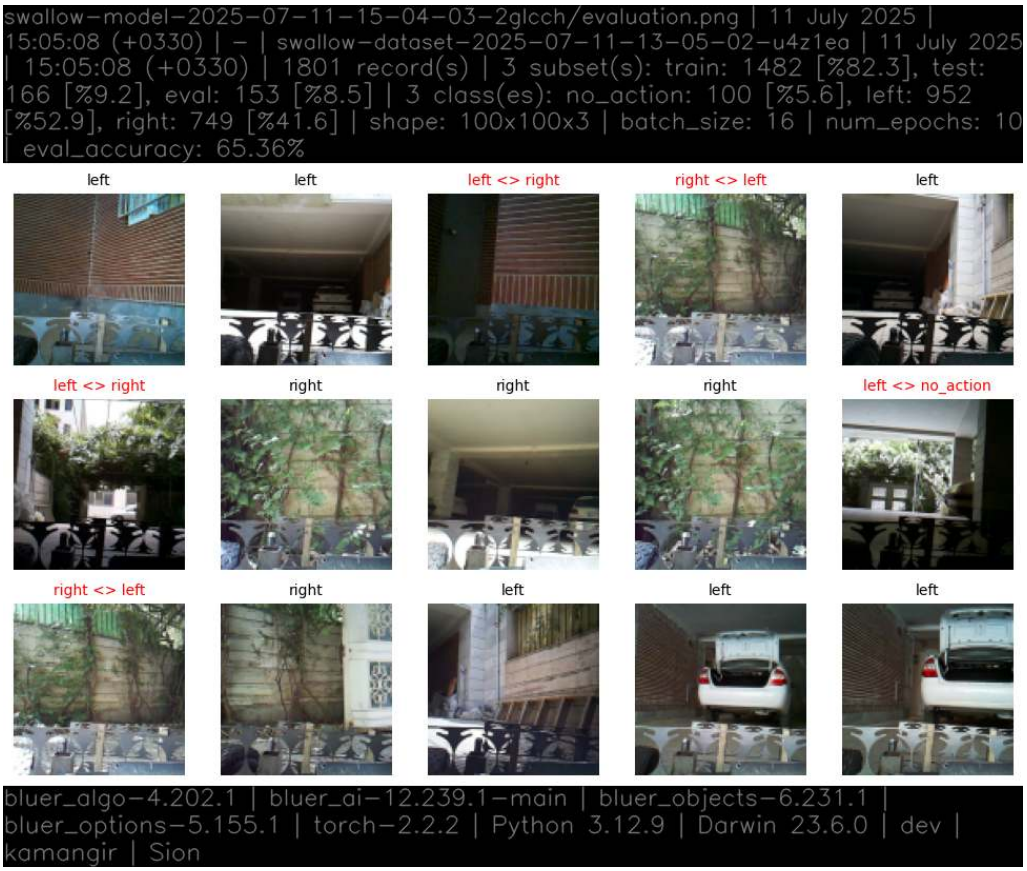
image

swallow-dataset-2025-07-11-13-05-02-u4z1ea

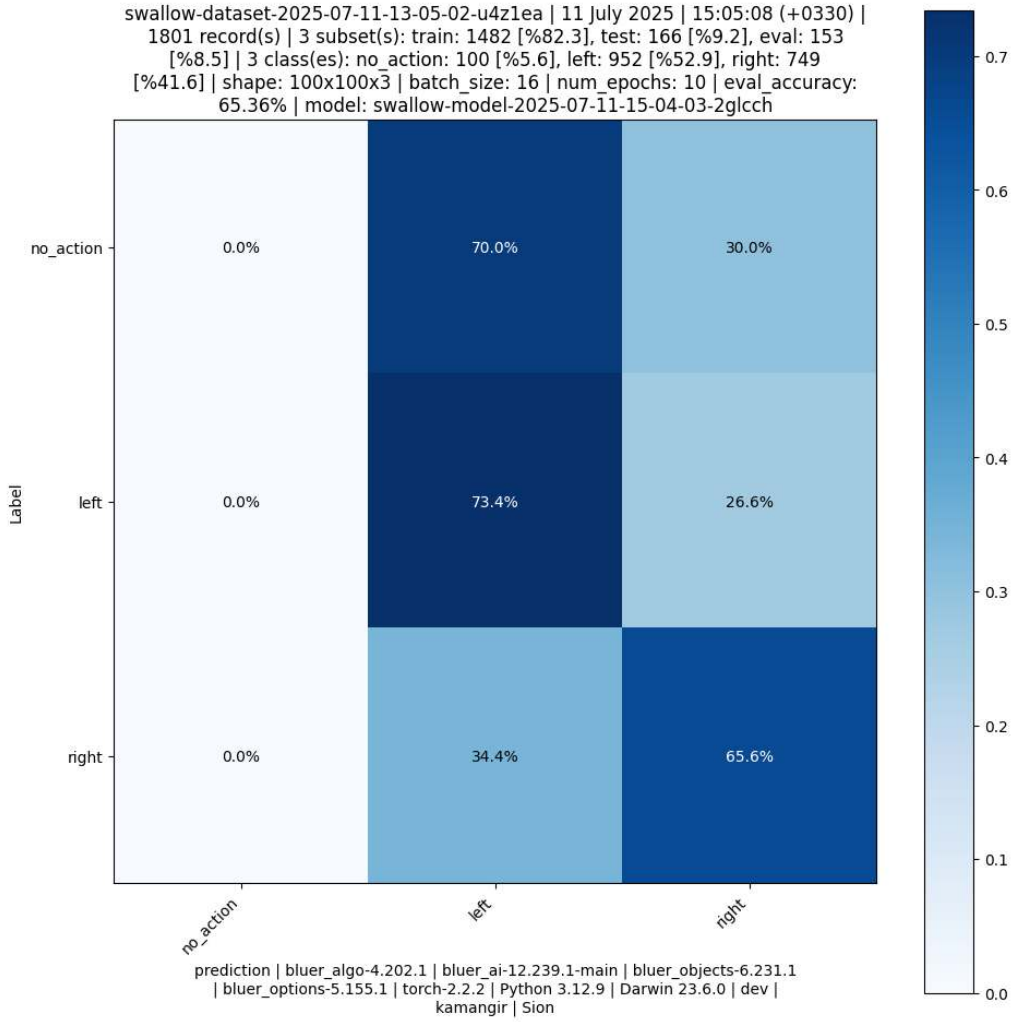
```
dataset:
  class_count: 3
  classes:
    0: no_action
    1: left
    2: right
  contains:
  - 2025-07-09-11-16-52-4zo4zc
  - 2025-07-09-11-34-19-bcoh75
  count: 1801
  shape:
  - 100
  - 100
  - 3
  subsets:
    eval: 153
    test: 166
    train: 1482
```



image


swallow-model-2025-07-11-15-04-03-2glcch/evaluation.png | 11 July 2025 | 15:05:08 (+0330) | - | swallow-dataset-2025-07-11-13-05-02-u4z1ea | 11 July 2025 | 15:05:08 (+0330) | 1801 record(s) | 3 subset(s): train: 1482 [%82.3], test: 166 [%9.2], eval: 153 [%8.5] | 3 class(es): no_action: 100 [%5.6], left: 952 [%52.9], right: 749 [%41.6] | shape: 100x100x3 | batch_size: 16 | num_epochs: 10 | eval_accuracy: 65.36%
left
left
left <> right
right <> left
left
left <> right
right
right
right
left <> no_action
right <> left
right
left
left
left
bluer_algo-4.202.1 | bluer_ai-12.239.1-main | bluer_objects-6.231.1 | bluer_options-5.155.1 | torch-2.2.2 | Python 3.12.9 | Darwin 23.6.0 | dev | kamangir | Sion

image



image

[swallow-model-2025-07-11-15-04-03-2glcch]

```
model:
  dataset:
    class_count: 3
    classes:
      0: no_action
      1: left
      2: right
    count: 1801
    shape:
    - 100
    - 100
    - 3
  evaluation:
    class_accuracy:
      0: 0.0
      1: 0.7341772151898734
      2: 0.65625
    eval_accuracy: 0.6535947712418301
  inputs:
    batch_size: 16
    num_epochs: 10
  training:
    loss:
    - 0.8941771462861343
    - 0.8678298002956045
    - 0.8598417815891838
```

- 0.863602487181845
- 0.8459089610740723
- 0.8423866080208186
- 0.8415681831588951
- 0.8320272445035206
- 0.8420564680286103
- 0.8336275264962643

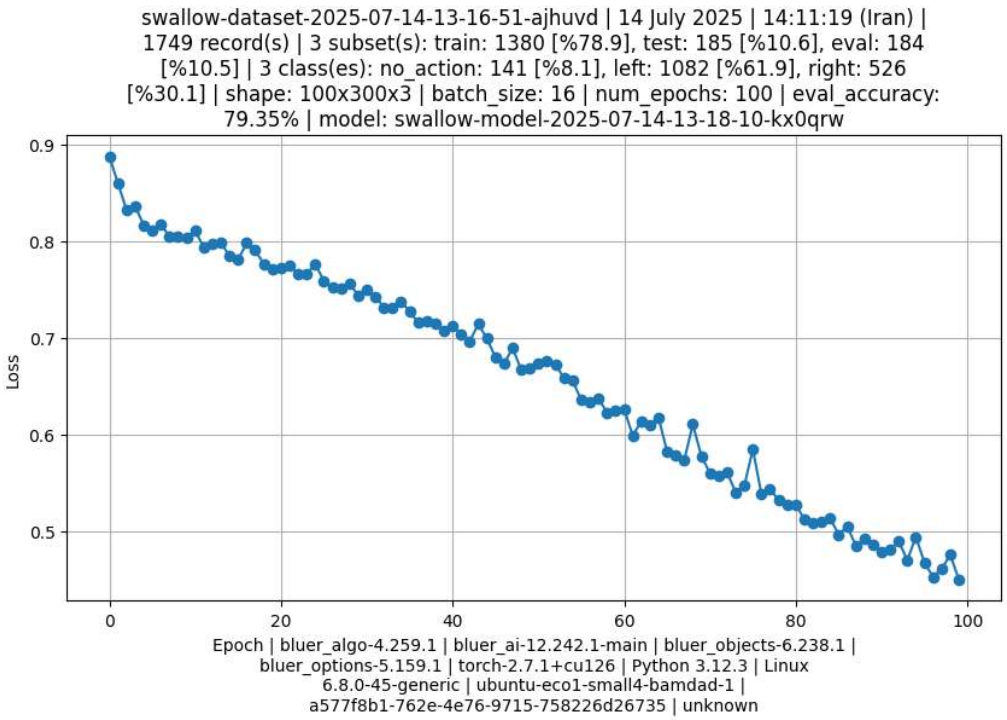
swallow: digital: algo: navigation: model: one

uses combination/one.

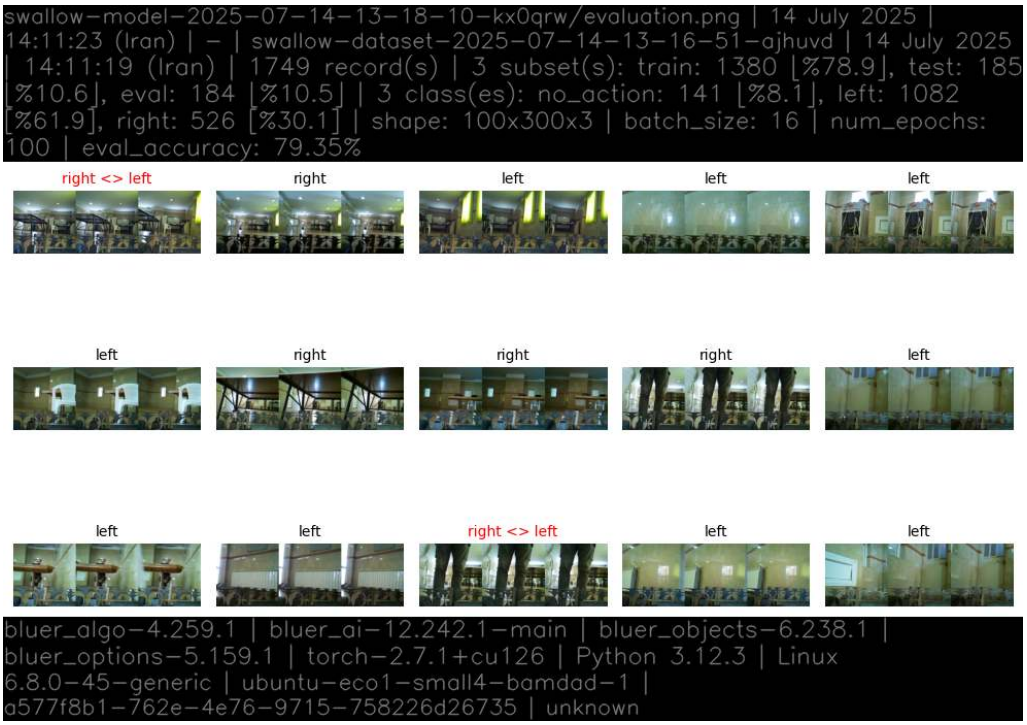
```
@arvan ssh <ip-address>
@arvan seed
# Ctrl+V

@select swallow-dataset-$(@timestamp)

@swallow dataset combine \
    sequence=3 . \
    --datasets $(@list filter \
        $(@swallow dataset list) \
        --contains 2025-07-13)

@upload filename=metadata.yaml .
@assets publish \
    extensions=png,push . \
    --prefix grid

@select swallow-model-$(@timestamp)

@image_classifier model train upload .. . \
    --num_epochs 100

@upload public,zip .
@assets publish \
    extensions=png,push .

@select swallow-prediction-test-$(@timestamp)

@algo image_classifier model prediction_test \
    upload ... .. .

@assets publish \
    extensions=png,push .
```

image

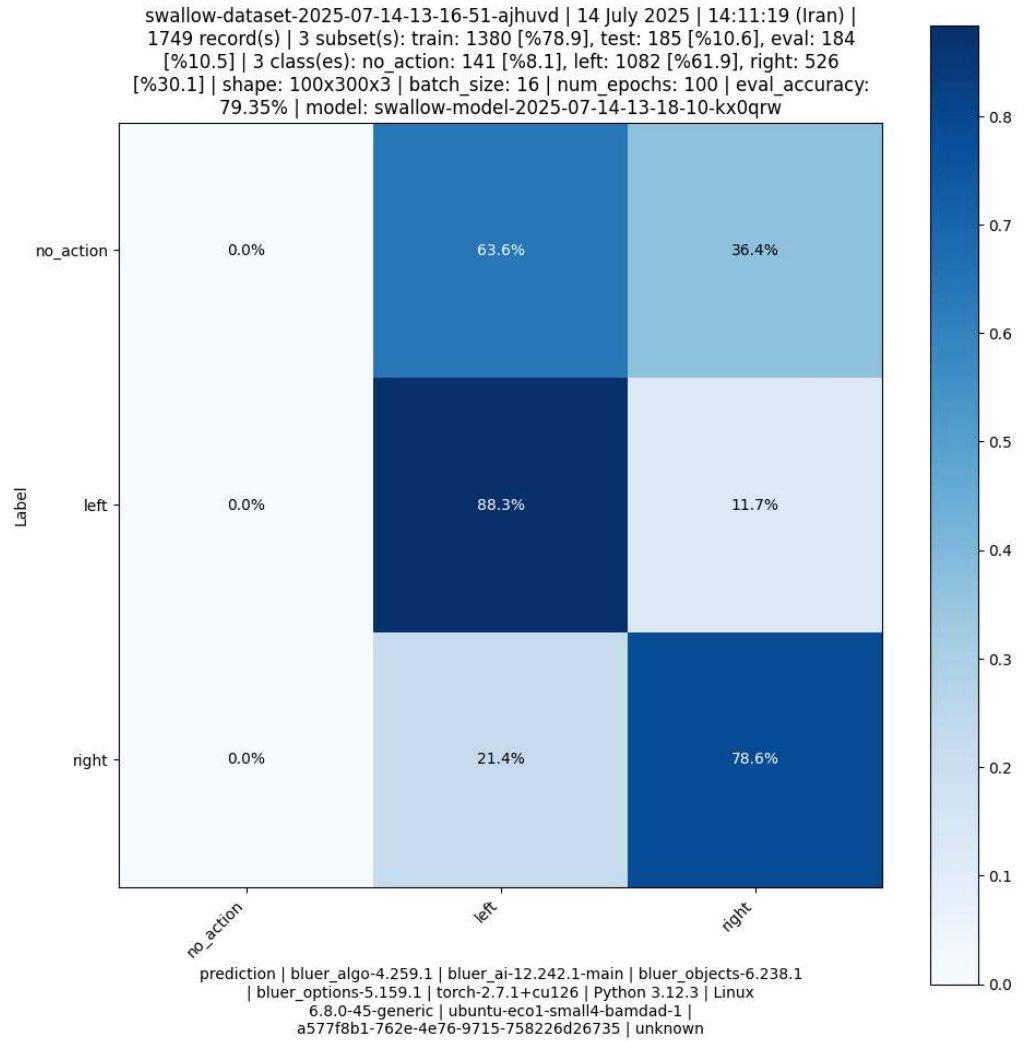
▶ metadata



image



image



image

[swallow-model-2025-07-14-13-18-10-kx0qrw]

▶ metadata

---

swallow-prediction-test-2025-07-14-14-13-57-ngywj1 | 14 July 2025 | 14:14:30 (Iran) | model: swallow-model-2025-07-14-13-18-10-kx0qrw | 3 class(es): #0: no_action, #1: left, #2: right | shape: 100x300x3 | prediction: right [#2] | correct | took 398 ms | bluer_algo-4.259.1 | bluer_ai-12.242.1-main | bluer_objects-6.238.1 | bluer_options-5.159.1 | torch-2.7.1+cu126 | Python 3.12.3 | Linux 6.8.0-45-generic | ubuntu-eco1-small4-bamdad-1 | a577f8b1-762e-4e76-9715-758226d26735 | unknown

image

▶ metadata

aliases: image-classifier

dataset

```
@image_classifier \
    dataset \
    ingest \
    [clone,count=<100>,source=fruits_360,upload] \
    [-|<object-name>] \
    [--class_count -1] \
    [--test_ratio 0.1] \
    [--train_ratio 0.8]
 . ingest -> <object-name>.
@image_classifier \
    dataset \
    review \
    [~download,upload] \
    [.|<object-name>]
 . review <object-name>.
@image_classifier \
    dataset \
    sequence \
    [~download,length=<2>,upload] \
    [.|<source-object-name>] \
    [-|<destination-object-name>]
 . <source-object-name> -sequence-> <destination-object-name>.
```

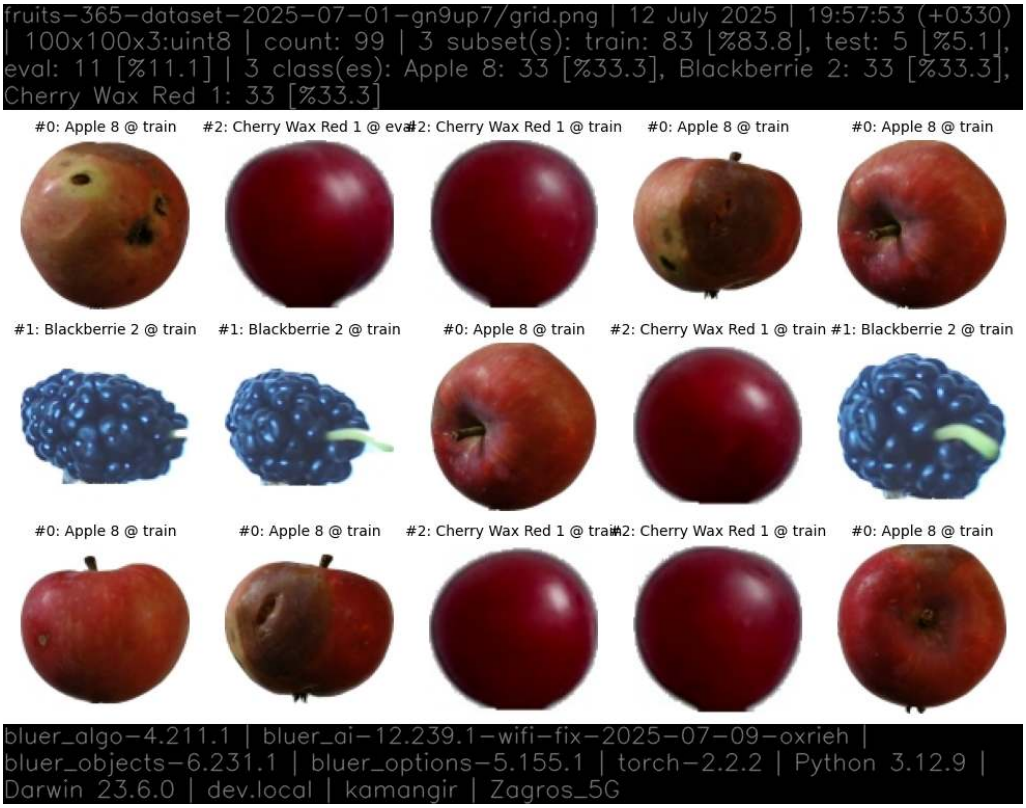
model

```
@image_classifier \
    model \
    prediction_test \
    [~download,upload] \
    [..|<dataset-object-name>] \
    [.|<model-object-name>] \
    [-|<prediction-object-name>]
 . test prediction.
@image_classifier \
    model \
    train \
    [~download,upload] \
    [.|<dataset-object-name>] \
    [-|<model-object-name>] \
    [--batch_size 16] \
    [--num_epochs 10]
 . train.
```

image-classifier: dataset: ingest

- continues https://github.com/kamangir/image-classifier-2.
- uses https://github.com/fruits-360/fruits-360-100x100

```
@select fruits-365-dataset-$(@@timestamp)

@algo image_classifier dataset ingest \
    clone,count=100,source=fruits_360,upload . \
    --class_count 3

@upload public,zip .
@assets publish \
    extensions=png,push .
```



image

[fruits-365-dataset-2025-07-01-gn9up7]

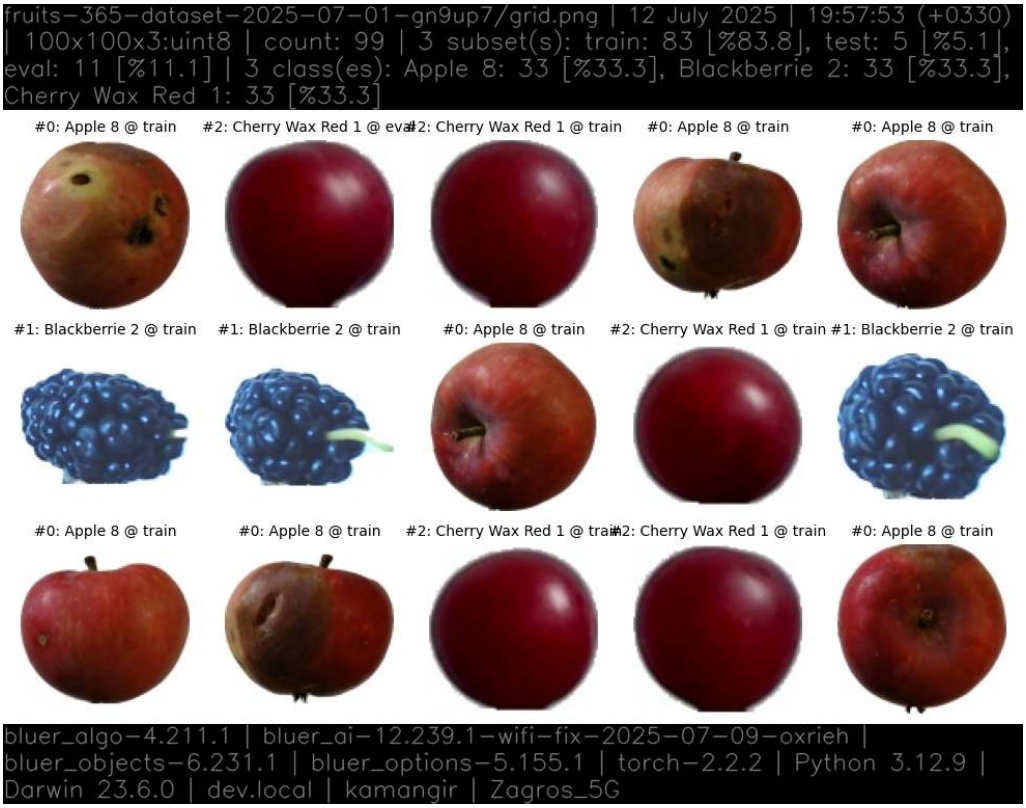
```
dataset:
  class_count: 3
  classes:
    0: Apple 8
    1: Blackberrie 2
    2: Cherry Wax Red 1
  count: 99
  ratios:
    eval: 0.09999999999999998
    test: 0.1
    train: 0.8
```

```
shape:
- 100
- 100
- 3
source: fruits_360
subsets:
  eval: 11
  test: 5
  train: 83
```

image-classifier: dataset: review

uses ingest.

```
@select $BLUER_ALGO_FRUITS_360_TEST_DATASET

@algo image_classifier dataset review - .

@upload public,zip .
@assets publish \
    extensions=png,push .
```



image

fruits-365-dataset-2025-07-01-gn9up7

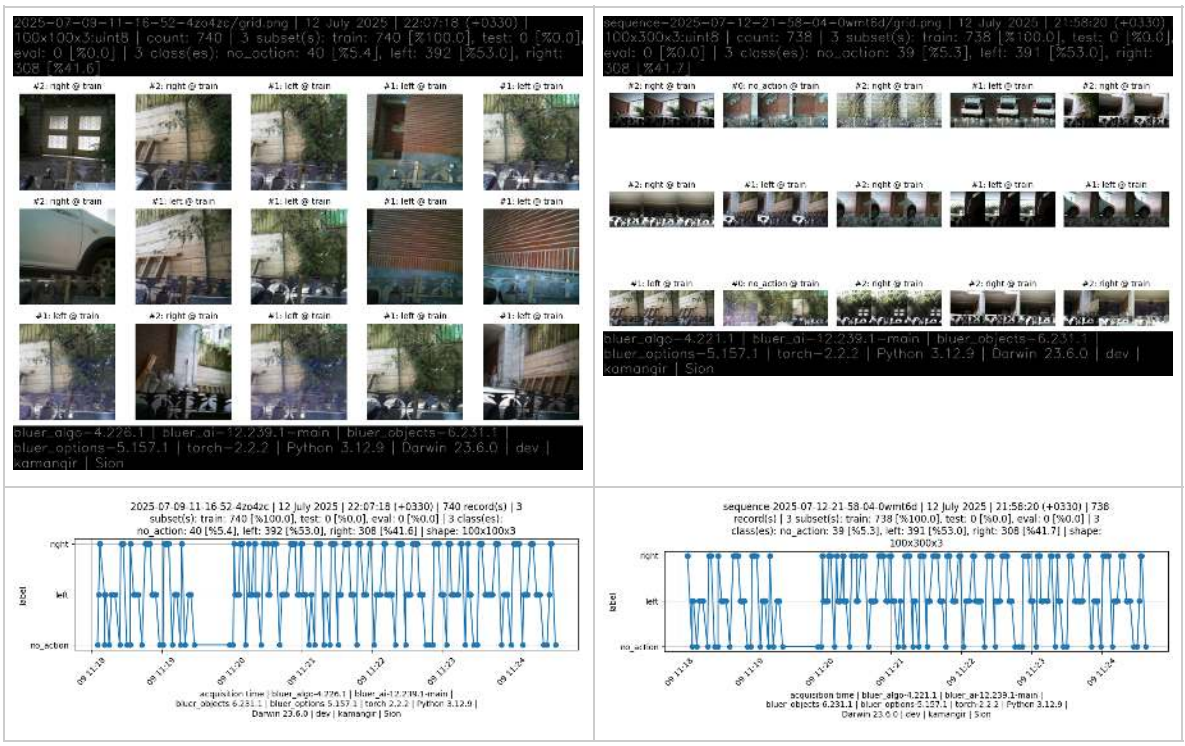
```
dataset:
  class_count: 3
  classes:
    0: Apple 8
    1: Blackberrie 2
    2: Cherry Wax Red 1
  count: 99
  ratios:
    eval: 0.09999999999999998
    test: 0.1
    train: 0.8
  shape:
  - 100
  - 100
```

```
- 3
source: fruits_360
subsets:
  eval: 11
  test: 5
  train: 83
```

image-classifier: dataset: sequence

uses bluer-ugv/swallow/digital/dataset/combination.

```
@select 2025-07-09-11-16-52-4zo4zc

@upload public,zip .
@assets publish \
    extensions=png,push . \
    --prefix grid

@select sequence-$(@timestamp)

@algo image_classifier dataset sequence \
    ~download,length=3 .. .

@upload public,zip .
@assets publish \
    extensions=png,push . \
    --prefix grid
```



[2025-07-09-11-16-52-4zo4zc](2025-07-09-11-16-52-4zo4zc)

```
dataset:
  class_count: 3
  classes:
    0: no_action
    1: left
    2: right
  count: 740
  shape:
```

```
  - 100
  - 100
  - 3
  source: 00000000c74cf7d2
  subsets:
    eval: 0
    test: 0
    train: 740
```

---

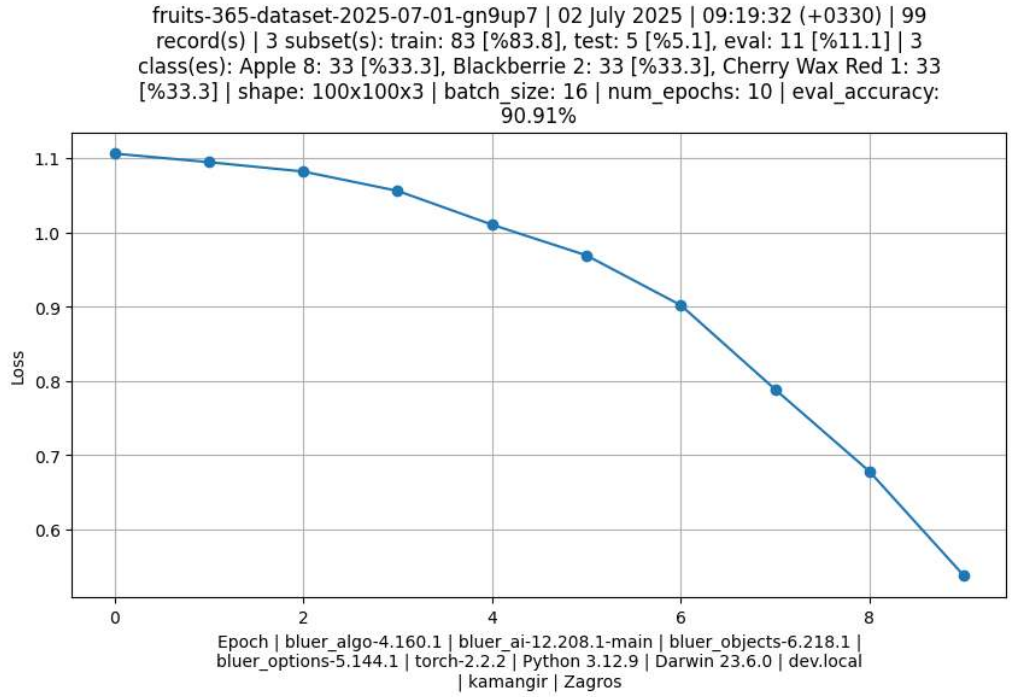
sequence-2025-07-12-21-58-04-0wmt6d

```
dataset:
  class_count: 3
  classes:
    0: no_action
    1: left
    2: right
  count: 738
  length: 3
  shape:
  - 100
  - 300
  - 3
  source: 2025-07-09-11-16-52-4zo4zc
  subsets:
    eval: 0
    test: 0
    train: 738
```

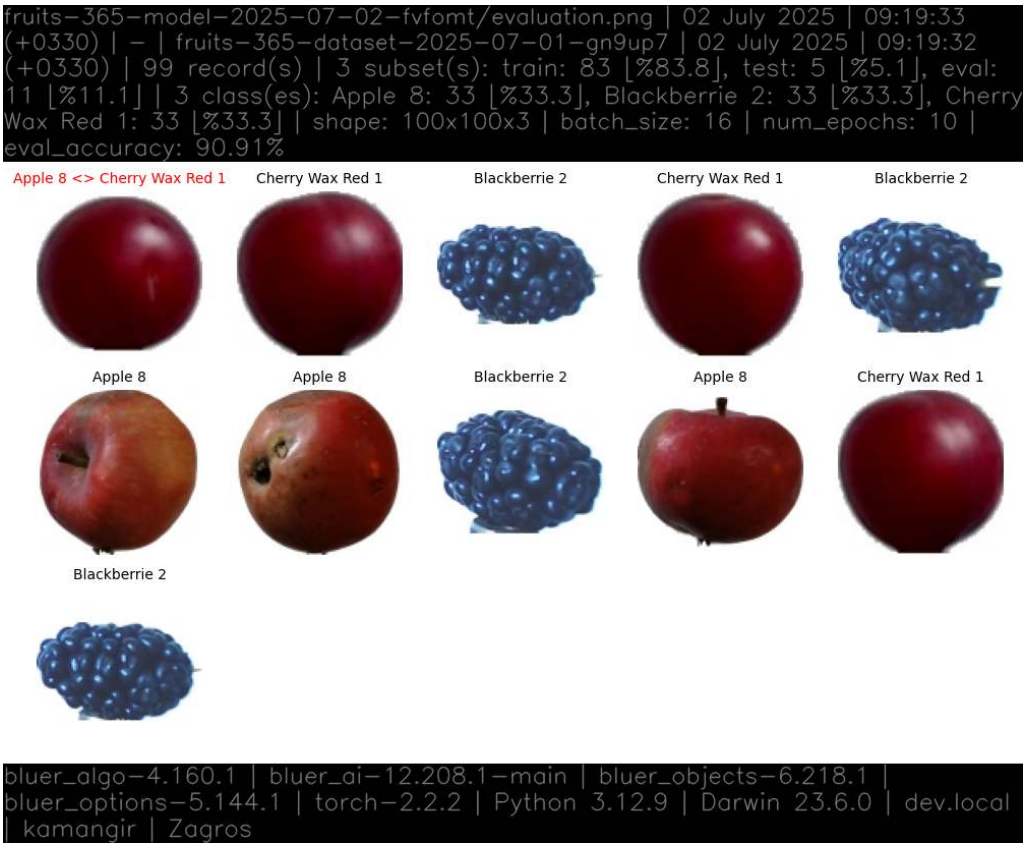
image-classifier: model: train: small

uses ingest.

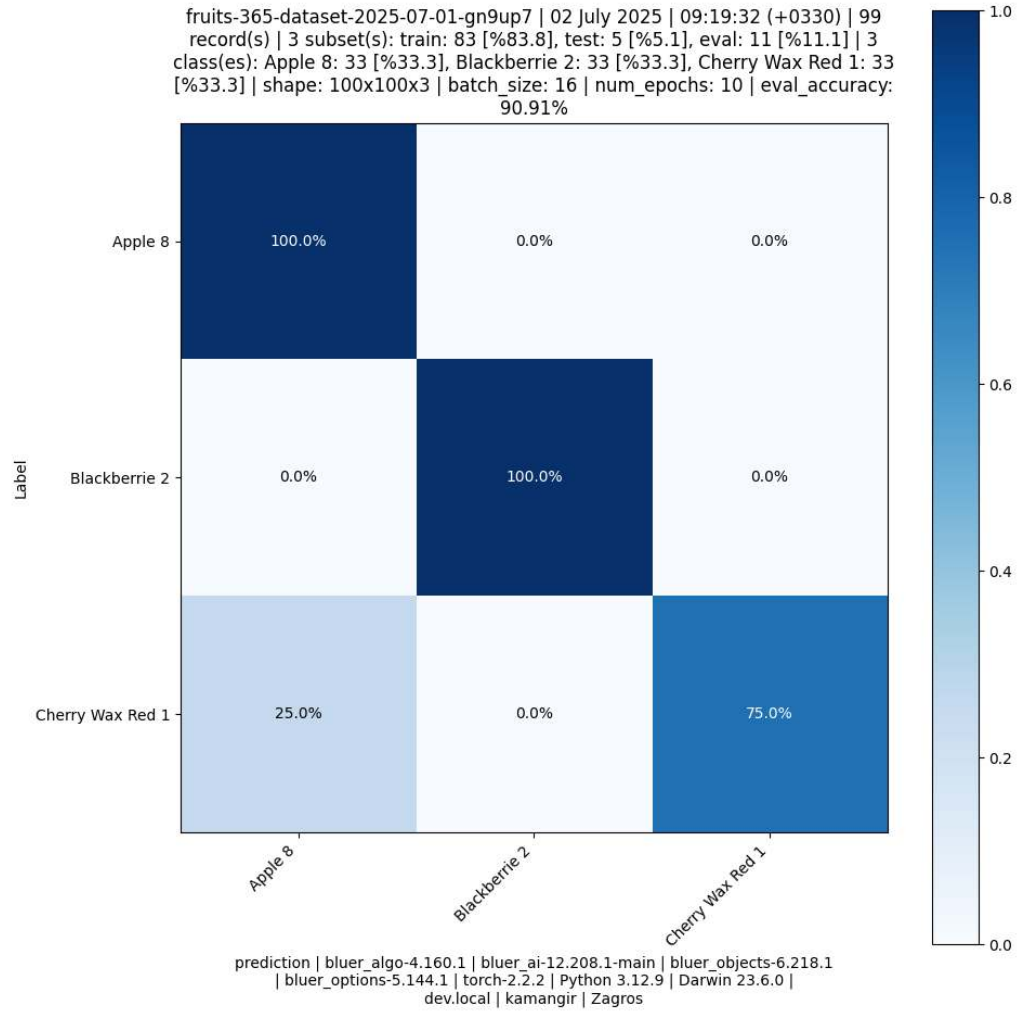
```
@select $BLUER_ALGO_FRUITS_360_TEST_DATASET
@select fruits-365-model-$(@@timestamp)

@algo image_classifier model train \
    upload .. .

@upload public,zip .
@assets publish \
    extensions=png,push .
```



image


fruits-365-model-2025-07-02-fvfomt/evaluation.png | 02 July 2025 | 09:19:33 (+0330) | - | fruits-365-dataset-2025-07-01-gn9up7 | 02 July 2025 | 09:19:32 (+0330) | 99 record(s) | 3 subset(s): train: 83 [%83.8], test: 5 [%5.1], eval: 11 [%11.1] | 3 class(es): Apple 8: 33 [%33.3], Blackberrie 2: 33 [%33.3], Cherry Wax Red 1: 33 [%33.3] | shape: 100x100x3 | batch_size: 16 | num_epochs: 10 | eval_accuracy: 90.91%
Apple 8 <> Cherry Wax Red 1
Cherry Wax Red 1
Blackberrie 2
Cherry Wax Red 1
Blackberrie 2
Apple 8
Apple 8
Blackberrie 2
Apple 8
Cherry Wax Red 1
Blackberrie 2
bluer_algo-4.160.1 | bluer_ai-12.208.1-main | bluer_objects-6.218.1 | bluer_options-5.144.1 | torch-2.2.2 | Python 3.12.9 | Darwin 23.6.0 | dev.local | kamangir | Zagros

image


fruits-365-dataset-2025-07-01-gn9up7 | 02 July 2025 | 09:19:32 (+0330) | 99 record(s) | 3 subset(s): train: 83 [%83.8], test: 5 [%5.1], eval: 11 [%11.1] | 3 class(es): Apple 8: 33 [%33.3], Blackberrie 2: 33 [%33.3], Cherry Wax Red 1: 33 [%33.3] | shape: 100x100x3 | batch_size: 16 | num_epochs: 10 | eval_accuracy: 90.91%
Label
Apple 8
Blackberrie 2
Cherry Wax Red 1
100.0%
0.0%
0.0%
0.0%
100.0%
0.0%
25.0%
0.0%
75.0%
Apple 8
Blackberrie 2
Cherry Wax Red 1
1.0
0.8
0.6
0.4
0.2
0.0
prediction | bluer_algo-4.160.1 | bluer_ai-12.208.1-main | bluer_objects-6.218.1 | bluer_options-5.144.1 | torch-2.2.2 | Python 3.12.9 | Darwin 23.6.0 | dev.local | kamangir | Zagros

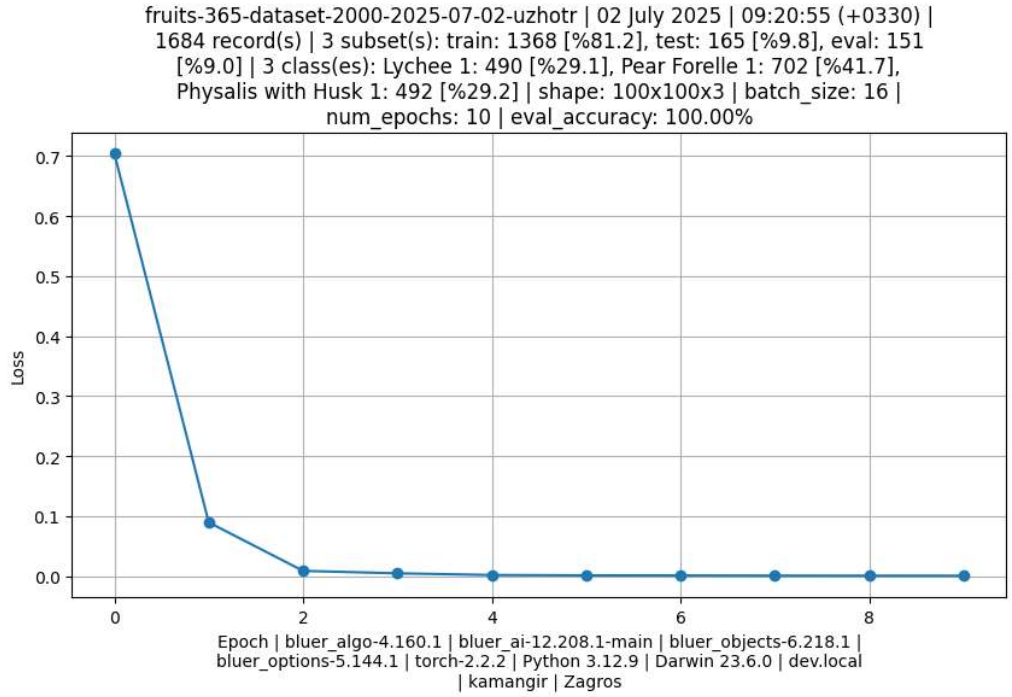
image

fruits-365-model-2025-07-02-fvfomt

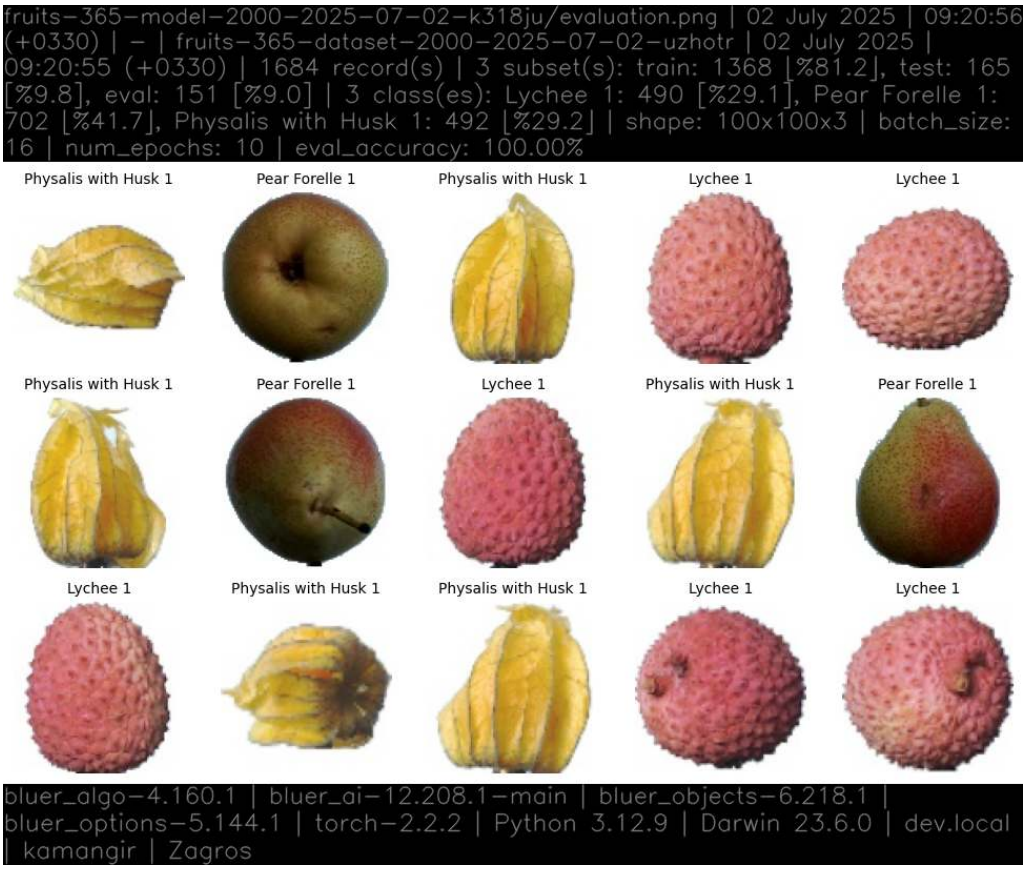

```
model:
  dataset:
    class_count: 3
    classes:
      0: Apple 8
      1: Blackberrie 2
      2: Cherry Wax Red 1
    count: 99
    shape:
    - 100
    - 100
    - 3
  evaluation:
    class_accuracy:
      0: 1.0
      1: 1.0
      2: 0.75
    eval_accuracy: 0.9090909090909091
  inputs:
    batch_size: 16
    num_epochs: 10
  training:
    loss:
    - 1.1059847007314842
    - 1.0944474734455707
    - 1.081830551825374
    - 1.0557171982454967
    - 1.0102007518331688
    - 0.9688218681209059
    - 0.9016210835382162
    - 0.7881300334470818
    - 0.6774013846753592
    - 0.5369924443313875
```

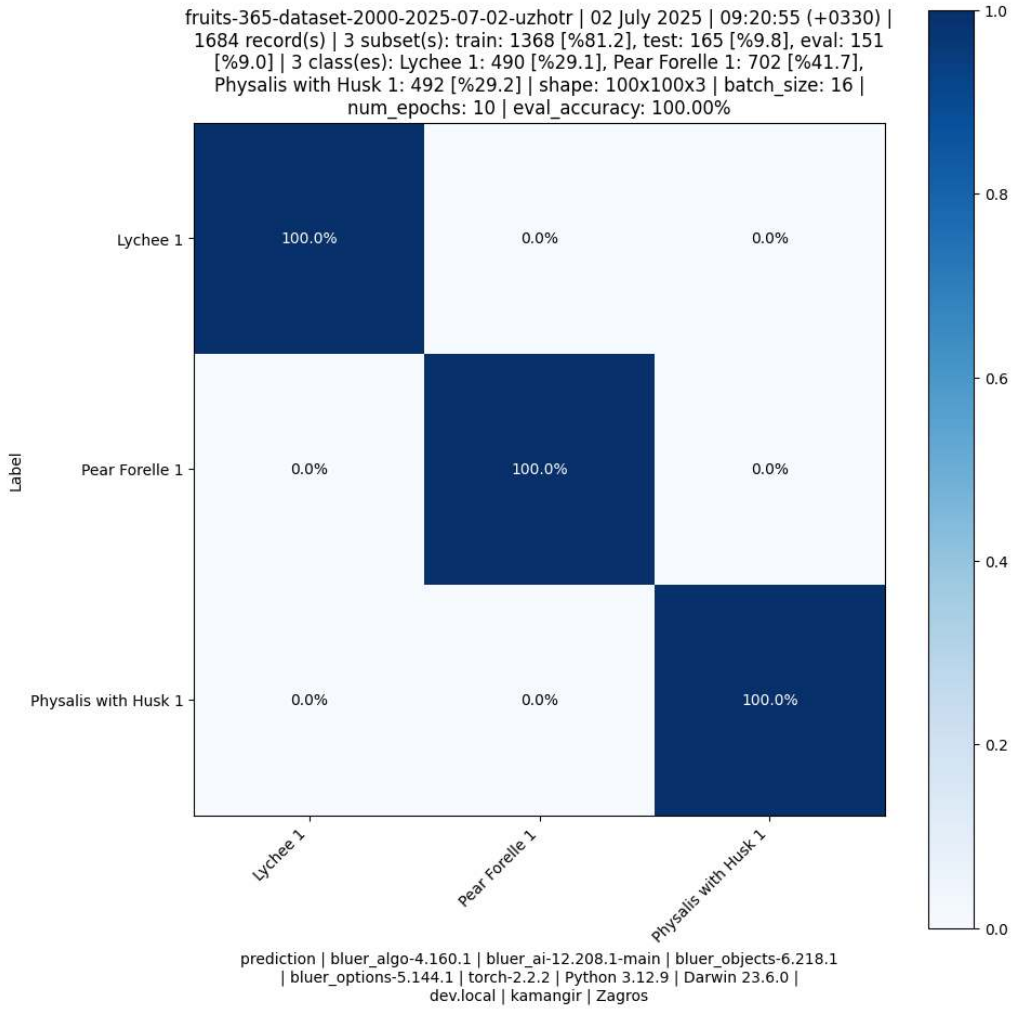
image-classifier: model: train: large

```
@select fruits-365-dataset-2000-$(@@timestamp)

@algo image_classifier dataset ingest \
    clone,count=10000,source=fruits_360 . \
    --class_count 3

@select fruits-365-model-2000-$(@@timestamp)

@algo image_classifier model train \
    ~download,upload .. .

@upload public,zip .
@assets publish \
    extensions=png,push .
```



image


fruits-365-model-2000-2025-07-02-k318ju/evaluation.png | 02 July 2025 | 09:20:56 (+0330) | - | fruits-365-dataset-2000-2025-07-02-uzhotr | 02 July 2025 | 09:20:55 (+0330) | 1684 record(s) | 3 subset(s): train: 1368 [%81.2], test: 165 [%9.8], eval: 151 [%9.0] | 3 class(es): Lychee 1: 490 [%29.1], Pear Forelle 1: 702 [%41.7], Physalis with Husk 1: 492 [%29.2] | shape: 100x100x3 | batch_size: 16 | num_epochs: 10 | eval_accuracy: 100.00%
Physalis with Husk 1
Pear Forelle 1
Physalis with Husk 1
Lychee 1
Lychee 1
Physalis with Husk 1
Pear Forelle 1
Lychee 1
Physalis with Husk 1
Pear Forelle 1
Lychee 1
Physalis with Husk 1
Physalis with Husk 1
Lychee 1
Lychee 1
bluer_algo-4.160.1 | bluer_ai-12.208.1-main | bluer_objects-6.218.1 | bluer_options-5.144.1 | torch-2.2.2 | Python 3.12.9 | Darwin 23.6.0 | dev.local | kamangir | Zagros
image



image

[fruits-365-model-2000-2025-07-02-k318ju]


```
model:
  dataset:
    class_count: 3
    classes:
      0: Lychee 1
      1: Pear Forelle 1
      2: Physalis with Husk 1
    count: 1684
    shape:
    - 100
    - 100
    - 3
  evaluation:
    class_accuracy:
      0: 1.0
      1: 1.0
      2: 1.0
    eval_accuracy: 1.0
  inputs:
    batch_size: 16
    num_epochs: 10
  training:
    loss:
    - 0.7043376307912738
    - 0.08943271208881286
    - 0.008551867894572343
```

- 0.004447948475223993
- 0.0016600372687074743
- 0.0010692106054555244
- 0.0008688547763543214
- 0.0006009176277234873
- 0.0004981696757456296
- 0.0004295692094449789

image-classifier: model: prediction: dev

uses train.

image_classifier_prediction.ipynb

image_classifier-prediction-2025-07-02-13-33-50-lgr380 | 02 July 2025 | 13:33:50 (+0330) | model: fruits-365-model-2025-07-02-fvfomt | 3 class(es): #0: Apple 8, #1: Blackberrie 2, #2: Cherry Wax Red 1 | shape: 100x100x3 | prediction: Apple 8 [#0] | correct | took 015 ms | bluer_algo-4.179.1 | bluer_ai-12.208.1 | bluer_objects-6.220.1 | bluer_options-5.144.1 | torch-2.2.2 | Python 3.12.9 | Darwin 23.6.0 | | Jupyter-Notebook

image

```
prediction:
  elapsed_time: 0.015111207962036133
  predicted_class: 0
```

image-classifier: model: prediction: rpi

uses train.

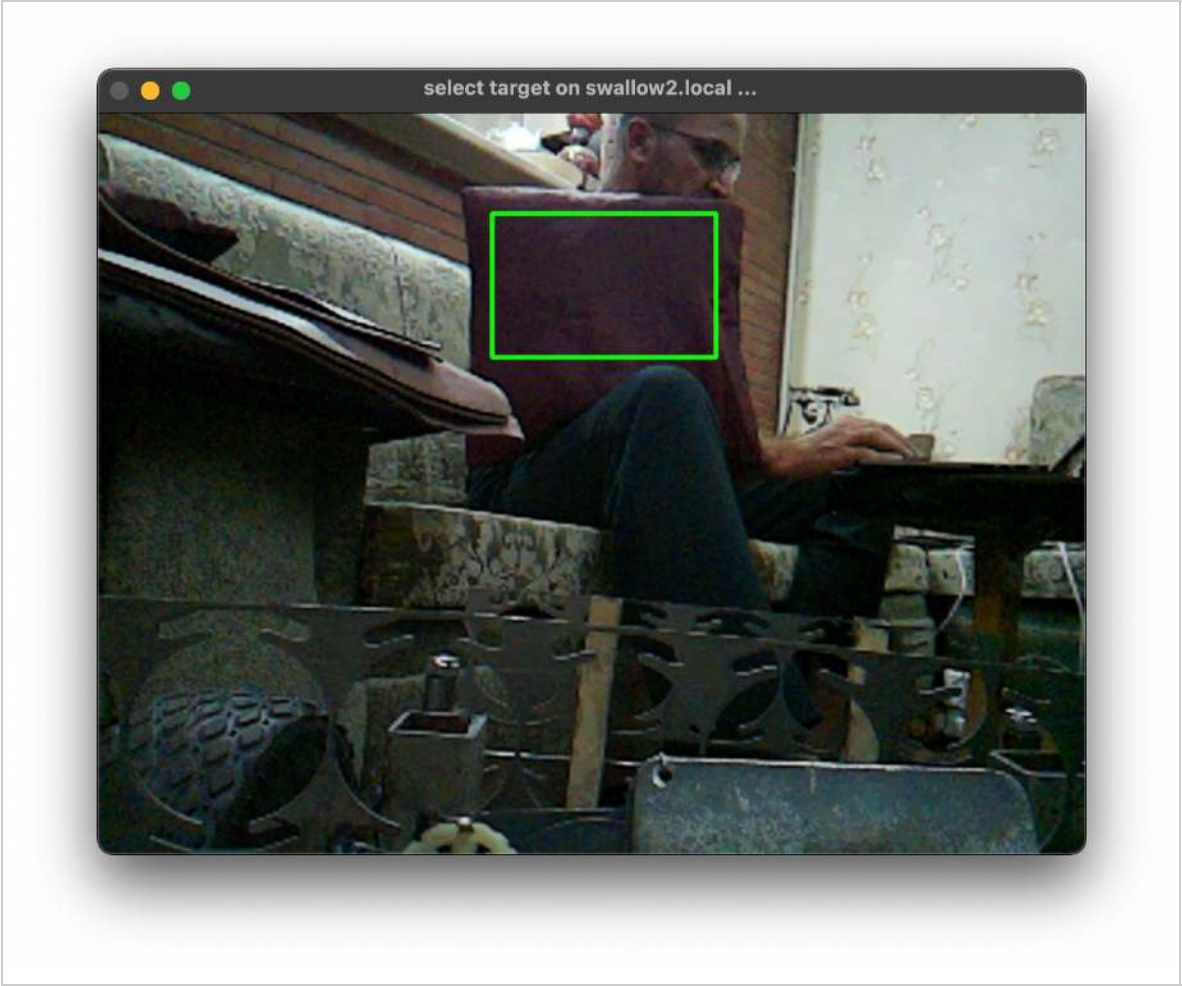
```
@algo test what=test_bluer_algo_image_classifier_model_prediction_test \
    upload
```

image_classifier-prediction-2025-07-02-16-11-03-0kb1d3 | 02 July 2025 | 16:11:39 (+0330) | model: fruits-365-model-2025-07-02-fvfomt | 3 class(es): #0: Apple 8, #1: Blackberrie 2, #2: Cherry Wax Red 1 | shape: 100x100x3 | prediction: Apple 8 [#0] | correct | took 081 ms | bluer_algo-4.185.1 | bluer_ai-12.208.1-main | bluer_objects-6.220.1 | bluer_options-5.139.1 | torch-2.4.1 | Python 3.8.10 | Linux 5.4.0-1129-raspi | swallow | 7p7yq | Sion

image

```
prediction:
  elapsed_time: 0.08102989196777344
  predicted_class: 0
```

swallow: digital: algo: tracking



target tracking using an @algo/tracker.

```
@swallow env cp tracking
```

aliases: tracker

```
@algo \
    tracker \
    [algo=camshift|meanshift,camera,~download,dryrun,sandbox,upload] \
    [-|<object-name>] \
    [--frame_count <10>] \
    [--log 1] \
    [--show_gui 1] \
    [--verbose 1]
 . run algo.
```

tracker

target tracking for a ugv using mean/cam-shift.

- sandbox: sandbox/mean-cam-shift
- camshift
- meanshift

> ℹ️ conclusion: `camshift` tracks distinct colors robustly and adjusts to object size.

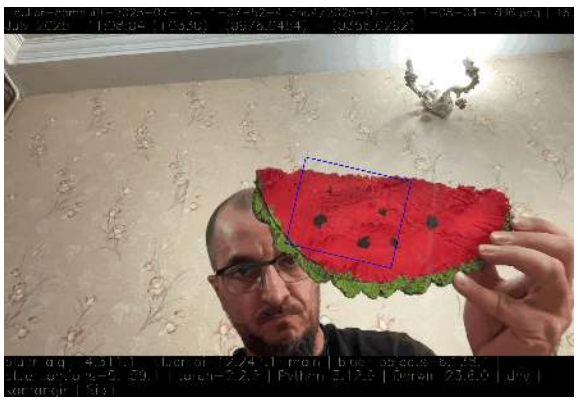
tracker: camshift

on a video file

```
@select tracker-camshift-$(@timestamp)

@algo tracker \
    algo=camshift . \
    --log 1

@assets publish \
    extensions=gif,push .
```

on camera feed

```
@select tracker-camshift-$(@timestamp)

@algo tracker \
    algo=camshift,camera . \
    --log 1 \
    --show_gui 1

@assets publish \
    extensions=gif,push .
```



tracker: meanshift

on a video file

```
@select tracker-meanshift-$(@timestamp)

@algo tracker \
    algo=meanshift . \
    --log 1

@assets publish \
    extensions=gif,push .
```

on camera feed

```
@select tracker-meanshift-$(@timestamp)

@algo tracker \
    algo=meanshift,camera . \
    --log 1 \
    --show_gui 1

@assets publish \
    extensions=gif,push .
```

swallow: digital: algo: yolo

target tracking using an @algo/yolo.

```
@swallow env cp yolo
```

- training a 256x256 model

```
@swallow debug
```

swallow: digital: algo: yolo: train

training a `256 x 256` model.

ingest

```
@select coco128-$(@@timestamp)

@yolo dataset ingest - . \
    --classes person

@yolo dataset review \
    ~download .

@upload public,zip

@assets publish \
    extensions=png,push .
```

coco128-2025-09-16-ko2aq9

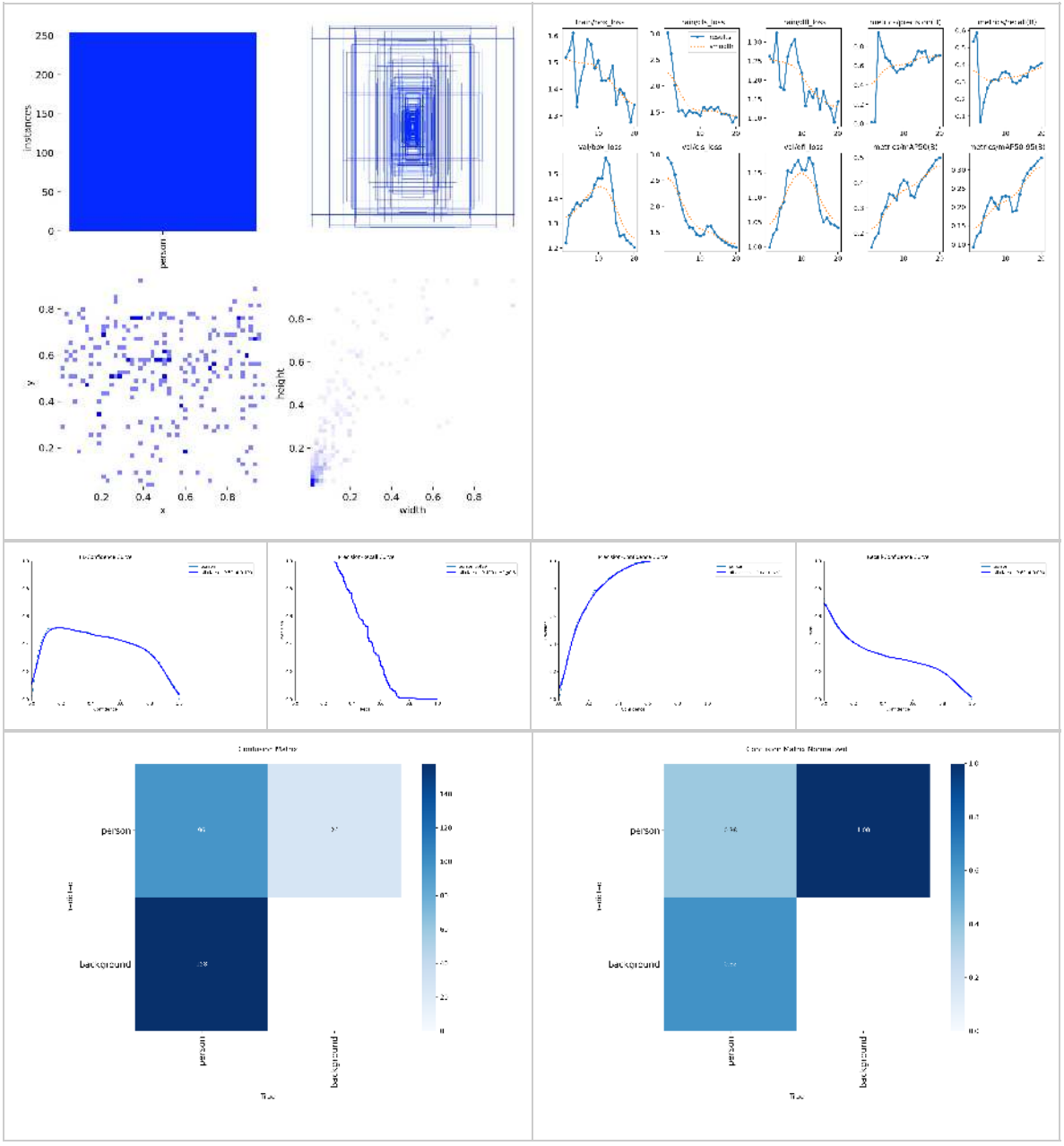
image

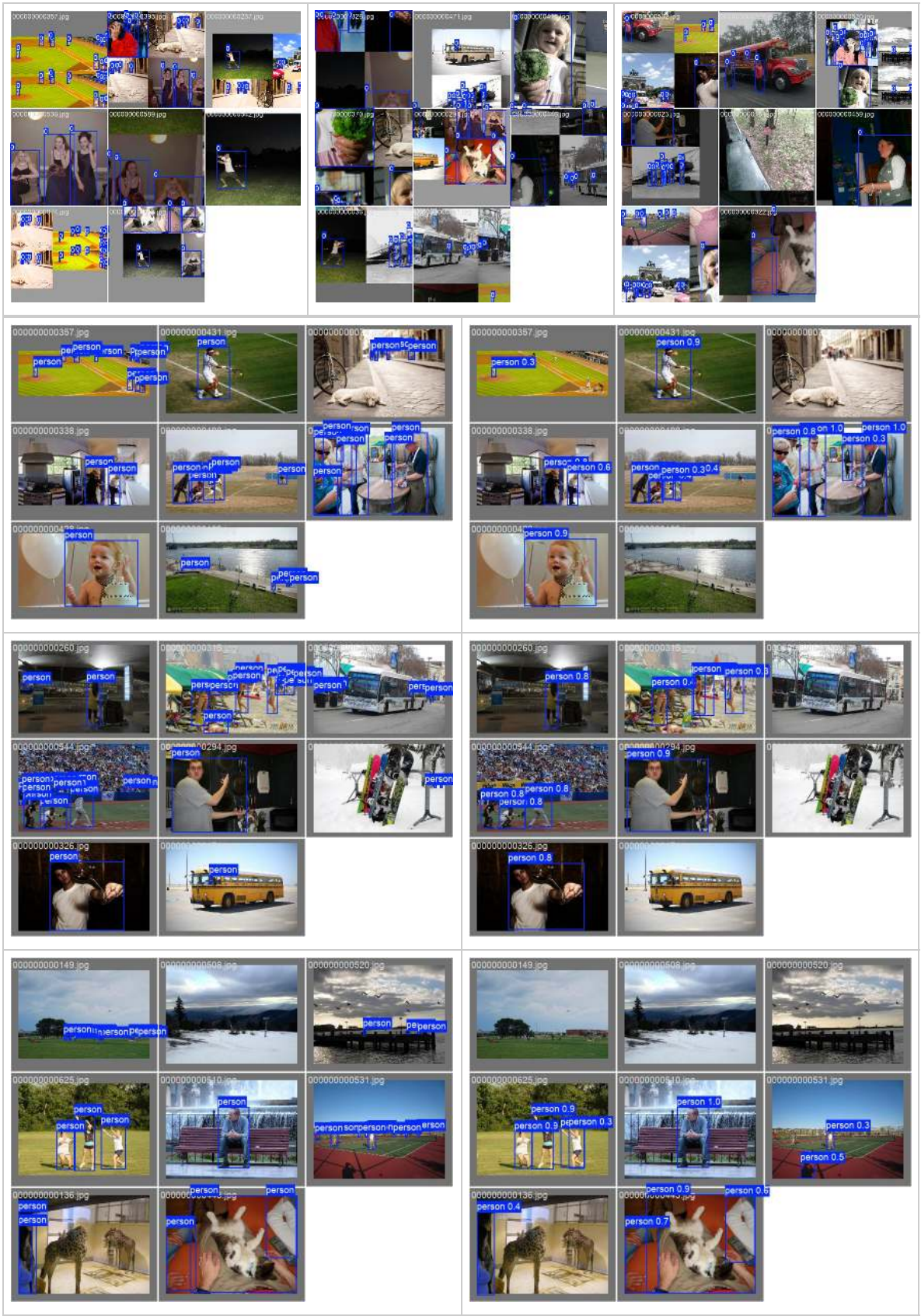
```
dataset:
  count: 61
names:
  0: person
source: coco_128
train: coco128/images/train2017
val: coco128/images/train2017
```

train

```
@select coco128-model-$(@@timestamp)

@yolo model train \
    ~download,upload .. . \
    --epochs 20 \
    --image_size 256

@upload public,zip

@assets publish \
    extensions=jpg+png . \
    --prefix train/

@assets publish \
    extensions=jpg+png,push . \
    --prefix validation/
```

coco128-model-2025-09-16-meb4if



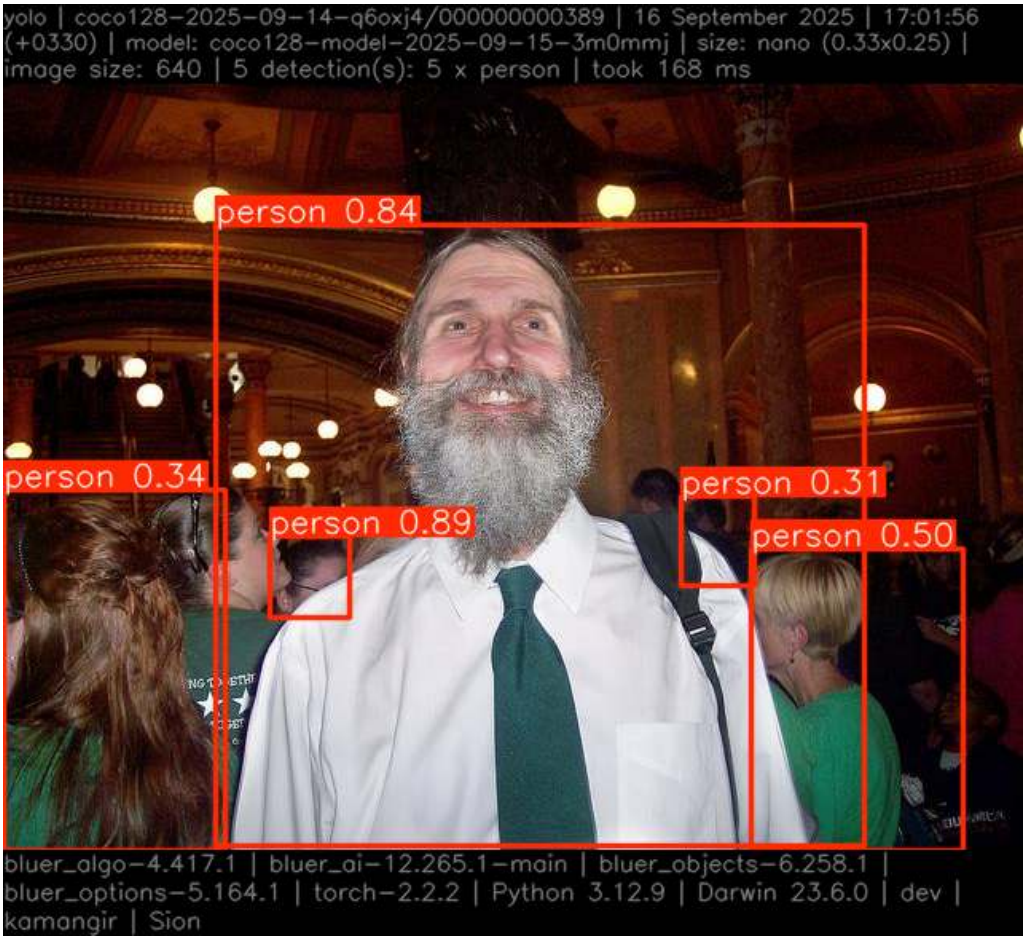


▶ metadata

predict

```
@select yolo-prediction-test-$(@timestamp)

@yolo model prediction_test \
```

```
    upload \
    $BLUER_ALGO_COCO128_TEST_DATASET \
    $BLUER_UGV_SWALLOW_YOLO_MODEL . \
    --record_index 3

@assets publish extensions=png,push
```



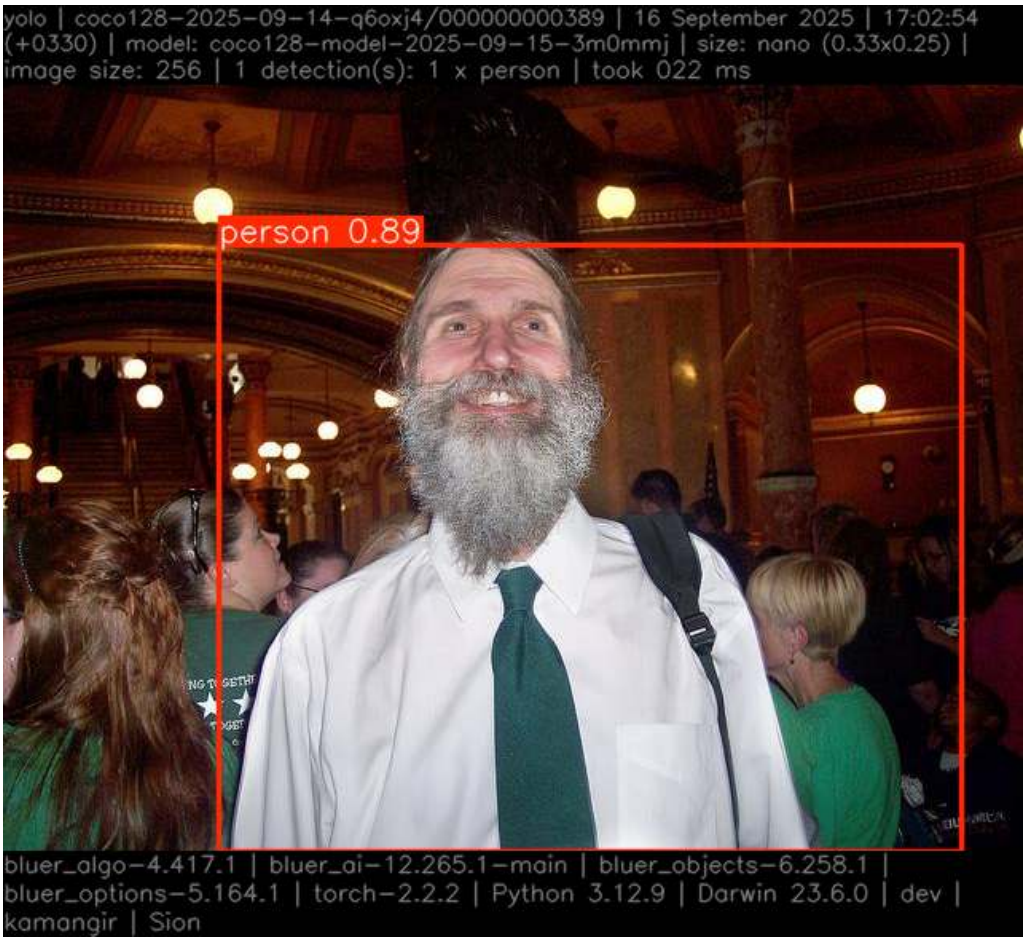
image

```
000000000389:
  detections:
  - bbox_xyxy:
    - 167.85377502441406
    - 283.70269775390625
    - 217.58558654785156
    - 334.81072998046875
    class_id: 0
    confidence: 0.8922585248947144
    label: person
  - bbox_xyxy:
    - 133.1409912109375
    - 88.43701171875
    - 540.6605834960938
    - 477.65869140625
    class_id: 0
    confidence: 0.842871367931366
    label: person
  - bbox_xyxy:
    - 469.0397644042969
    - 291.564697265625
    - 602.921875
    - 480.0
    class_id: 0
```

```
    confidence: 0.49844595789909363
    label: person
  - bbox_xyxy:
    - 0.58984375
    - 254.95266723632812
    - 139.97897338867188
    - 480.0
    class_id: 0
    confidence: 0.33697929978370667
    label: person
  - bbox_xyxy:
    - 425.3193359375
    - 258.2589416503906
    - 471.283935546875
    - 314.4782409667969
    class_id: 0
    confidence: 0.31127116084098816
    label: person
  elapsed_time: 0.16768479347229004
  image_size:
  - 480
  - 640
```

predict (256)

```
@select yolo-prediction-test-$(@timestamp)

@yolo model prediction_test \
    upload \
    $BLUER_ALGO_COCO128_TEST_DATASET \
    $BLUER_UGV_SWALLOW_YOLO_MODEL . \
    --record_index 3 \
    --image_size 256

@assets publish extensions=png,push
```

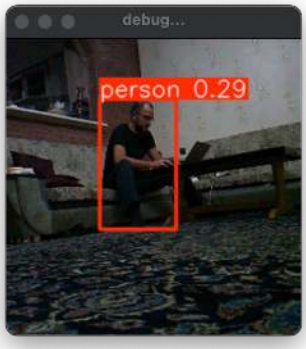


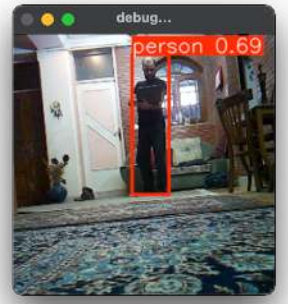
image


```
000000000389:
  detections:
  - bbox_xyxy:
    - 135.89393615722656
    - 100.03055572509766
    - 601.81640625
    - 480.0
    class_id: 0
    confidence: 0.8894820213317871
    label: person
  elapsed_time: 0.022478818893432617
  image_size:
  - 480
  - 640
```


predict (rpi)

```
@select swallow-debug-$(@timestamp)

@swallow debug .

@assets publish extensions=gif,push
```


debug...
person 0.29


debug...
person 0.69


debug...
person 0.88


debug...
person 0.95


debug...
person 0.87

aliases: yolo

dataset

```
@yolo \
    dataset \
    ingest \
    [dryrun,source=coco_128,upload] \
    [-|<object-name>] \
    [--classes all | person+boat] \
    [--verbose 1]
 . ingest -> <object-name>.
@yolo \
    dataset \
    review \
    [~download,upload] \
    [.|<object-name>] \
    [--verbose 1]
 . review <object-name>.
```

model

```
@yolo \
    model \
    prediction_test \
    [~download,upload] \
    [..|<dataset-object-name>] \
    [.|<model-object-name>] \
    [-|<prediction-object-name>] \
    [--image_size 640] \
    [--record_index 0] \
    [--warmup 0]
 . test prediction.
@yolo \
    model \
    train \
    [~download,upload] \
    [.|<dataset-object-name>] \
    [-|<model-object-name>] \
    [--batch 8] \
    [--device cpu | 0 | 0,1] \
    [--epochs 30] \
    [--from_scratch 1] \
    [--image_size 640] \
    [--model_size nano | small | medium | large | xlarge] \
    [--validate 0] \
    [--verbose 1] \
    [--workers 4]
 . train.
```

yolo: dataset: ingest-and-review

full

```
@select coco128-$(@@timestamp)

@yolo dataset ingest \
    upload .

@yolo dataset review \
    ~download .

@upload public,zip .
@assets publish \
    extensions=png,push .
```

image

[coco128-2025-09-14-q6oxj4]

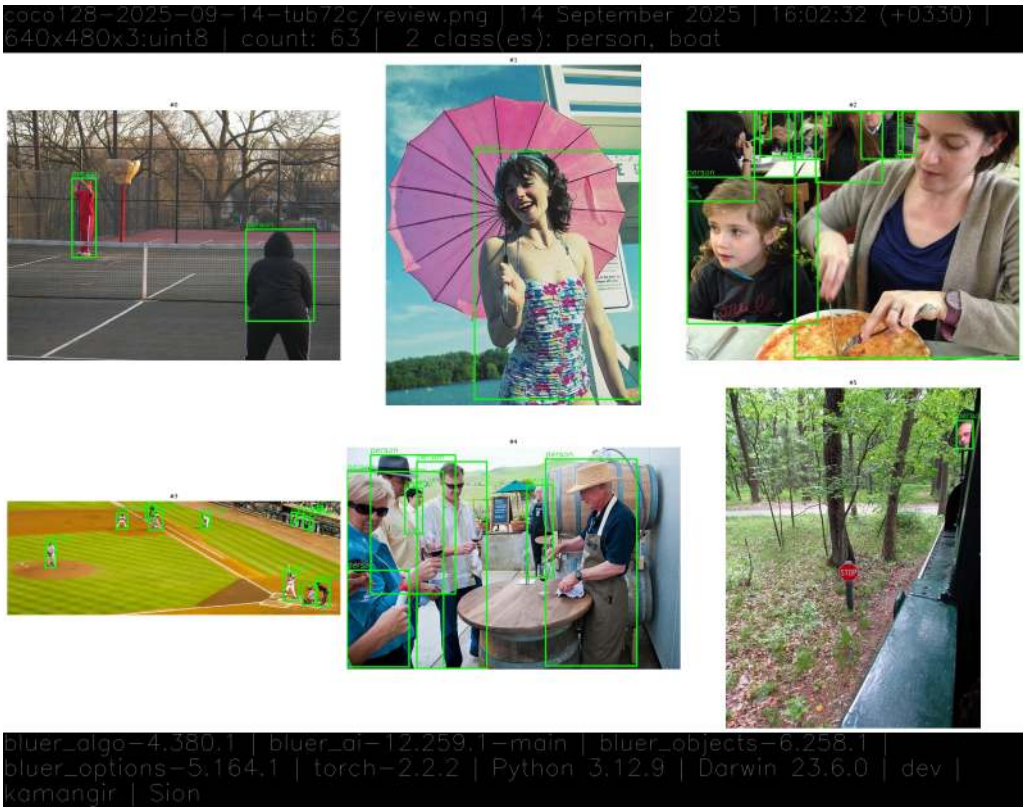
```
dataset:
  count: 126
names:
  0: person
  1: bicycle
  2: car
  3: motorcycle
```

```
4: airplane
5: bus
6: train
7: truck
8: boat
9: traffic light
10: fire hydrant
11: stop sign
12: parking meter
13: bench
14: bird
15: cat
16: dog
17: horse
18: sheep
19: cow
20: elephant
21: bear
22: zebra
23: giraffe
24: backpack
25: umbrella
26: handbag
27: tie
28: suitcase
29: frisbee
30: skis
31: snowboard
32: sports ball
33: kite
34: baseball bat
35: baseball glove
36: skateboard
37: surfboard
38: tennis racket
39: bottle
40: wine glass
41: cup
42: fork
43: knife
44: spoon
45: bowl
46: banana
47: apple
48: sandwich
49: orange
50: broccoli
51: carrot
52: hot dog
53: pizza
54: donut
55: cake
56: chair
57: couch
58: potted plant
59: bed
60: dining table
61: toilet
62: tv
63: laptop
64: mouse
65: remote
66: keyboard
67: cell phone
68: microwave
```

```
  69: oven
  70: toaster
  71: sink
  72: refrigerator
  73: book
  74: clock
  75: vase
  76: scissors
  77: teddy bear
  78: hair drier
  79: toothbrush
source: coco_128
train: coco128/images/train2017
val: coco128/images/train2017
```

select classes

```
@select coco128-$(@@timestamp)

@yolo dataset ingest \
    upload . \
    --classes person+boat

@yolo dataset review \
    ~download .

@upload public,zip .
@assets publish \
    extensions=png,push .
```



image

[coco128-2025-09-14-tub72c]

```
dataset:
  count: 63
```

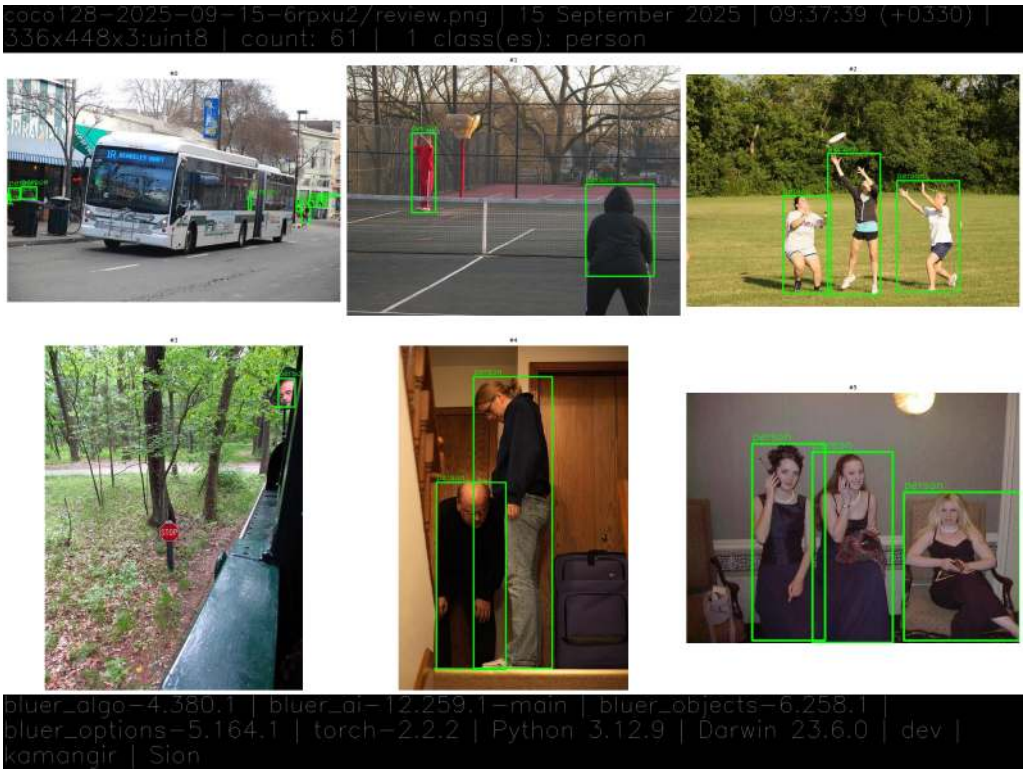
```
names:
  0: person
  1: boat
source: coco_128
train: coco128/images/train2017
val: coco128/images/train2017
```

yolo: model: validation

ingest

```
@select coco128-$(@@timestamp)

@yolo dataset ingest - . \
    --classes person

@yolo dataset review \
    ~download .

@upload public,zip

@assets publish \
    extensions=png,push .
```

coco128-2025-09-15-6rpxu2



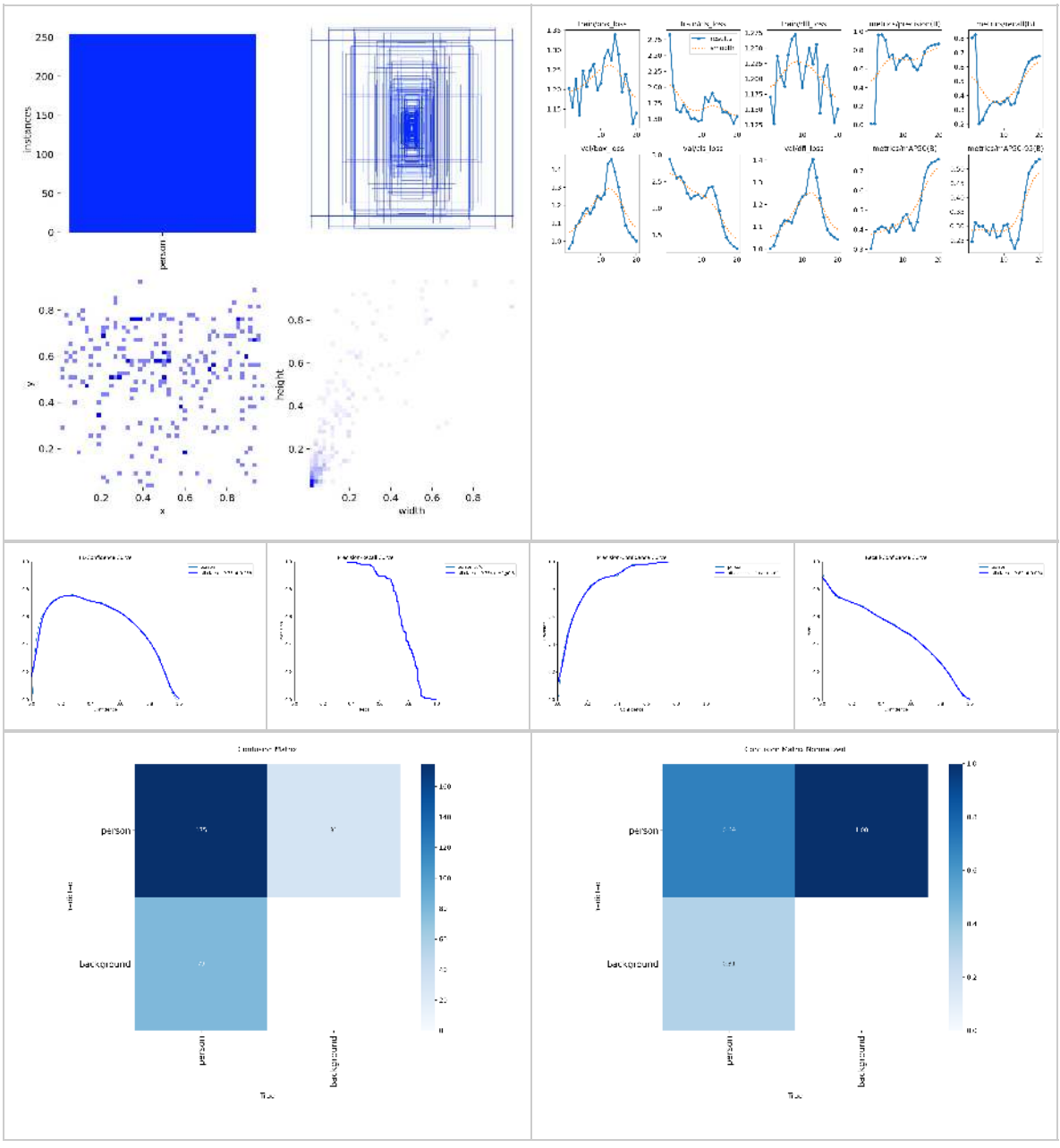
image

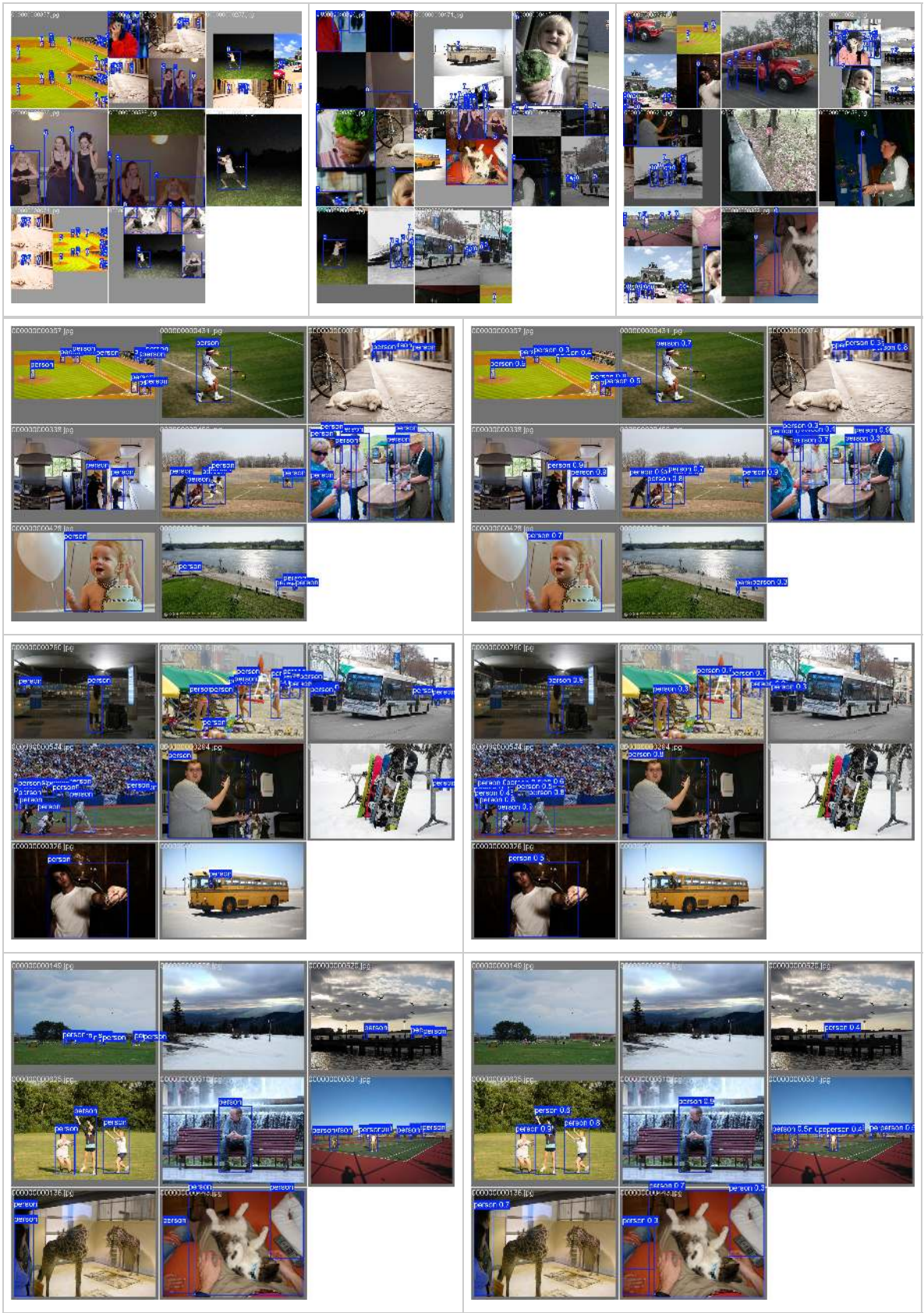
```
dataset:
  count: 61
names:
  0: person
source: coco_128
train: coco128/images/train2017
val: coco128/images/train2017
```

train

```
@select coco128-model-$(@@timestamp)

@yolo model train \
    ~download,upload .. . \
    --epochs 20

@upload public,zip

@assets publish \
    extensions=jpg+png . \
    --prefix train/

@assets publish \
    extensions=jpg+png,push . \
    --prefix validation/
```

[coco128-model-2025-09-15-3m0mmj]





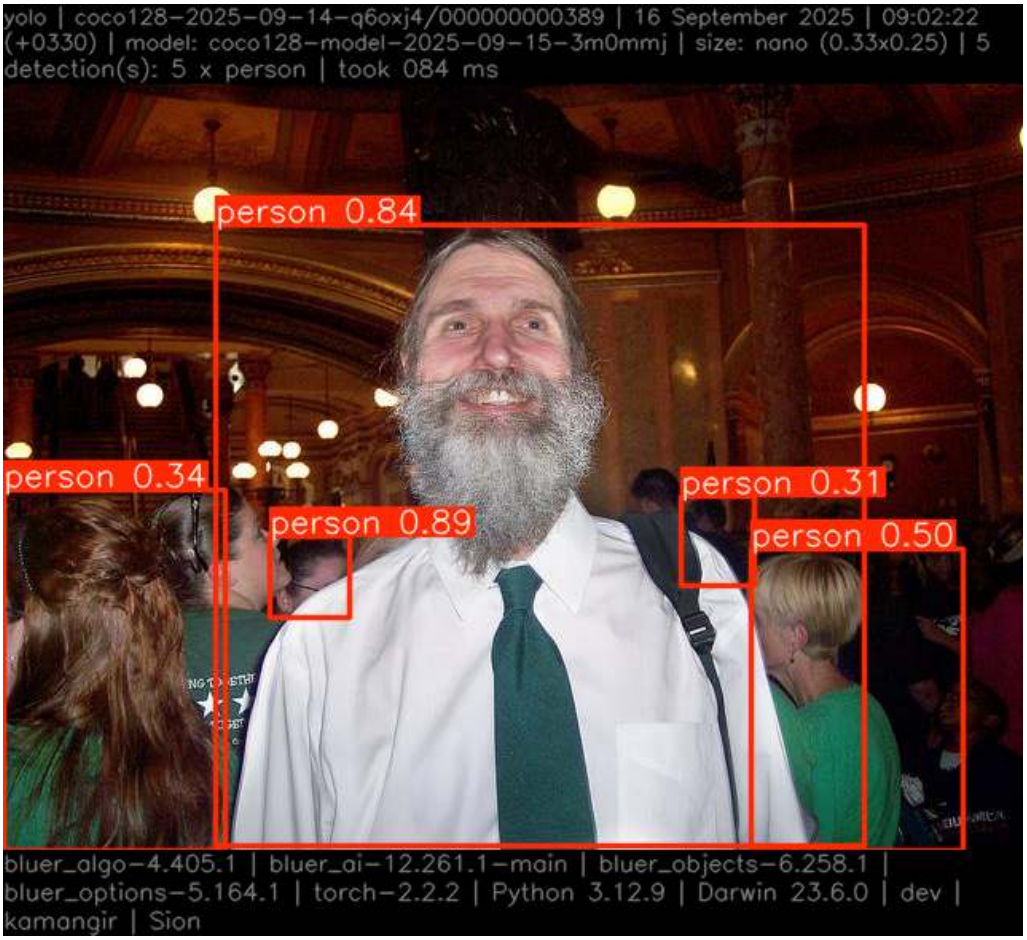
▶ metadata

predict (dev)

[yolo_prediction.ipynb]

predict

```
@select yolo-prediction-test-$(@timestamp)

@yolo model prediction_test \
    upload \
    $BLUER_ALGO_COCO128_TEST_DATASET \
    $BLUER_ALGO_COCO128_TEST_MODEL . \
    --record_index 3

@assets publish extensions=png,push
```



image

```
000000000389:
  detections:
  - bbox_xyxy:
    - 167.85377502441406
    - 283.70269775390625
    - 217.58558654785156
    - 334.81072998046875
    class_id: 0
    confidence: 0.8922585248947144
    label: person
  - bbox_xyxy:
    - 133.1409912109375
    - 88.43701171875
    - 540.6605834960938
    - 477.65869140625
    class_id: 0
    confidence: 0.842871367931366
    label: person
```

```
  - bbox_xyxy:
    - 469.0397644042969
    - 291.564697265625
    - 602.921875
    - 480.0
    class_id: 0
    confidence: 0.49844595789909363
    label: person
  - bbox_xyxy:
    - 0.58984375
    - 254.95266723632812
    - 139.97897338867188
    - 480.0
    class_id: 0
    confidence: 0.33697929978370667
    label: person
  - bbox_xyxy:
    - 425.3193359375
    - 258.2589416503906
    - 471.283935546875
    - 314.4782409667969
    class_id: 0
    confidence: 0.31127116084098816
    label: person
  elapsed_time: 0.08387088775634766
  image_size:
  - 480
  - 640
```


predict (rpi)

```
@select yolo-prediction-test-$(@timestamp)

@yolo model prediction_test \
    upload \
    $BLUER_ALGO_COCO128_TEST_DATASET \
    $BLUER_ALGO_COCO128_TEST_MODEL . \
    --record_index 3

@assets publish extensions=png,push
```

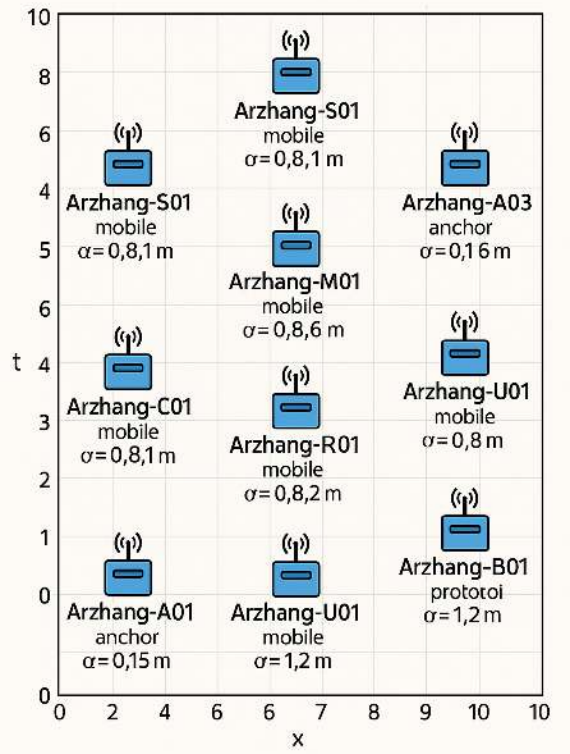
image

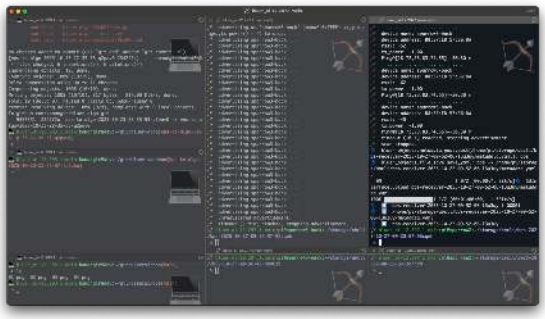

```
000000000419:
  detections:
  - bbox_xyxy:
    - 124.66294860839844
    - 121.88188934326172
    - 170.0297393798828
    - 281.0235595703125
    class_id: 0
    confidence: 0.7116779088973999
    label: person
  - bbox_xyxy:
    - 455.524658203125
    - 231.17758178710938
    - 635.5400390625
    - 478.5752868652344
    class_id: 0
    confidence: 0.4342321455478668
    label: person
  elapsed_time: 2.339571714401245
  image_size:
  - 480
  - 640
```

bps

bluer-positioning system.

> A distributed, serverless positioning mesh where every Raspberry Pi acts as both a tag and a local estimator: each node continuously advertises a compact BLE packet containing its node ID, mobility flag, and its current best-estimate pose (x,y,σ,seq), and simultaneously scans for neighbor adverts to collect RSSI + neighbor poses. Anchors are marked static with known coordinates and low uncertainty; mobiles fuse anchors + neighbor pseudo-ranges (RSSI→distance, calibrated per site) with a lightweight weighted least-squares + EKF/Covariance-Intersection step, adapt advertising/scan rates when moving, and use short rolling HMAC tokens to prevent simple spoofing. The result is a gossiping swarm that converges locally to consistent positions, degrades gracefully to coarse proximity when RF is noisy, and can be migrated later to UWB without changing the peer-to-peer protocol.





ℹ️ works on rpi only.

ℹ️ tx-power is not implemented in rpi. nominal value: 10-12 dBm. -1: indicates unknown.

aliases: bps

```
@bps \
    beacon \
    [~start_bluetooth] \
    [-|<object-name>] \
    [--generate 1] \
    [--sigma <4.0>] \
    [--simulate 1] \
    [--spacing <2.0>] \
    [--timeout <10.0 | -1>] \
    [--x <1.0>] \
    [--y <2.0>] \
    [--z <3.0>]
 . start beacon.
@bps \
    generate \
    [-] \
    [-|<object-name>] \
    [--as_str <x>,<y>,<z>,sigma] \
    [--only_validate 1] \
    [--sigma <4.0>] \
    [--simulate 1] \
    [--x <1.0>] \
    [--y <2.0>] \
    [--z <3.0>]
 . generate a ping.
@bps \
    install
 . install bps.
@bps \
    introspect \
    [~start_bluetooth,verbose,unique_bus_name=<1:234>]
 . introspect <1:234>.
@bps \
    loop \
    start \
    [simulate,upload] \
    [-|<object-name>]
 . start bps loop.
@bps \
    loop \
    stop \
    [rpi,wait] \
    [<machine-name>]
 . stop bps loop.
@bps \
    receiver \
    [~start_bluetooth,upload,verbose] \
    [-|<object-name>] \
    [--grep <sparrow+swallow>] \
    [--timeout <10>]
 . start receiver.
@bps \
    receiver \
    [~python,~start_bluetooth,verbose]
 . start receiver.
@bps \
```

```
    review \
    [~download,upload] \
    [.|<object-name>]
 . review <object-name>.
@bps \
    set_anchor \
    clear | <1.1,2.2,3.3,4.4>
 . set bps anchor.
@bps \
    simulate \
    timing \
    [upload] \
    [-|<object-name>] \
    [--length <1200>] \
    [--anchors <4>] \
    [--nodes <3>] \
    [--ta1 <20>] \
    [--ta2 <40>] \
    [--tr1 <20>] \
    [--tr2 <40>] \
    [--verbose 1]
 . simulate timing.
@bps \
    start_bluetooth \
    [verbose]
 . start bluetooth.
@bps \
    test \
    [~start_bluetooth,verbose]
 . d-bus ping test.
```

bps: literature

- Research Progress of Wireless Positioning Methods Based on RSSI - 2024

  A modern survey of RSSI-based positioning techniques across BLE, Wi-Fi, ZigBee, and 5G. It explains theoretical limitations of RSSI (path-loss variability, multipath fading) and outlines correction strategies.

- Bluetooth indoor positioning system based on improved RSSI data - 2024

- Peer-To-Peer UWB Ranges as a Source of Training Data for Estimating BLE RSSI Path-Loss Exponents - 2022

  Proposes using precise UWB peer-to-peer ranges to calibrate BLE path-loss parameters automatically. Shows how multi-modal measurements improve BLE localization accuracy by up to 40%.

- A Survey of Robot Swarms' Relative Localization Methods - 2022

  A comprehensive review of how robot swarms estimate relative positions using sensors such as BLE, UWB, infrared, and vision. It categorizes algorithms by communication type, sensing range, and scalability.

- From Robot Self-Localization to Global-Localization: An RSSI Based Approach" (ArXiv) - 2021

  presents a **simulation-based** method for enabling groups of mobile robots to localize themselves in GPS-denied environments. Some robots act as fixed beacons, broadcasting signals whose received strength (RSSI) is used by other robots to estimate their global position **through geometric reconstruction**. The approach was tested only in simulation (Webots) and not on physical robots. Reported average errors were about 0.6 m with 10 % Gaussian noise, 1.4 m with 20 % noise, and ≈0.1 m in near-ideal short-range conditions (3–3.5 m from beacons); within about 6 m of beacons, the total error remained under 1 m.

- Bluetooth Low Energy Technology Applied to Indoor Positioning Systems: An Overview - 2020

- RSSI Filtering Methods Applied to Localization using Bluetooth Low Energy - 2020

  Focuses on the signal-processing side of BLE localization. It evaluates Kalman filters, moving averages, and histogram-based smoothing to mitigate RSSI noise.

- Indoor localization using radio beacon technology - 2018

- Precise Realtime Indoor Localization With Raspberry Pi And Ultra-Wideband Technology (Decawave DWM1001 Developement Boards)

bps: mathematics: timing

optimizing advertise/receive cycle timing - continues v1.

problem setup

each machine runs the following infinite loop:

1. **advertise** for t_a seconds
2. **receive** (scan) for a random duration $U(t_{r1}, t_{r2})$ seconds

each process, advertise ($x = a$) or receive ($x = r$), takes:

- t_{xo} seconds to open
- t_{xc} seconds to close

define the total per-phase overhead:

$t_{as} = t_{ao} + t_{ac}$

$t_{rs} = t_{ro} + t_{rc}$

goal: Choose t_a, t_{r1}, and t_{r2} to maximize the expected number of advertisements received among all machines.

---

derivation

total cycle duration:

$T = t_a + t_{as} + t_{rm} + t_{rs}$

where,

$$ t_{rm} = \frac{1}{2}(t_{r1} + t_{r2}) : \text{mean receive time} $$

for many unsynchronized machines, the expected overlap fraction between “me listening” and “others advertising” is proportional to:

$$ P \propto \frac{t_a}{T} * \frac{t_{rm}}{T} $$

optimal relationship

for a fixed T,

$t_a + t_{rm} = T - (t_{as} + t_{rs})$

is fixed and P is maximized when,

$$ t_a = t_{rm} = \frac{1}{2}(T - t_{as} - t_{rs}) $$

$$ P \propto \frac{1}{4} \left(1 - \frac{t_{as} + t_{rs}}{T}\right)^2 $$

therefore larger T is more optimal for higher overlap.

the expected reaction time is,

$$ R \propto T - t_a = \frac{1}{2}(T + t_{as} + t_{rs}) $$

therefore, smaller T is more optimal for faster reaction.

summary

- match advertise and mean receive times: $t_a = (t_{r1} + t_{r2})/2$
- keep both much longer than $t_{as} + t_{rs}$
- add 10–30% jitter to t_{r1}/t_{r2} to avoid synchronization collisions
- choose smaller t_a to for faster reaction.

practical choices

system parameters,

t_{ao}	t_{ac}	t_{as}	t_{ro}	t_{rc}	t_{rs}
~1 s	~0 s	~1 s	~3 s	~0.5 s	~ 3s

chose:

- t_a = 30 s
- r_{r1} = 24 s
- r_{r2} = 36 s
- jitter: 20 %
- T: 64 s
- P: ~ 22%
- R: 34 s

bps: mathematics: localization

BLE Localization Cost Function with σ

This document defines the mathematical model used to estimate the position of a UGV based on BLE advertisements.
Each beacon advertises its position (x_i, y_i, z_i) and uncertainty σ_i along with its transmitted power.

---

1. RSSI–Distance Model

The received signal strength indicator (RSSI) is related to distance by the log-distance path-loss model:

$$\text{RSSI}_i = P_{\text{tx},\,i} - 10\, n_i \log_{10}(d_i) + \varepsilon_i$$

where:

- $P_{\text{tx},\,i}$ is the transmit power (dBm) of beacon i
- n_i is the path-loss exponent (≈ 2 for open space, higher indoors)
- ε_i is zero-mean Gaussian noise with standard deviation $\sigma_{\text{rssi},\,i}$

Rearranging gives an estimate of the distance to beacon i:

$$d_i = 10^{(P_{\text{tx},\,i} - \text{RSSI}_i)/(10\, n_i)}$$

---

2. Measurement Model

Let the UGV's unknown position be

$$\mathbf{p} = (x, y, z)$$

Each beacon i has a known position

$$\mathbf{p}_i = (x_i, y_i, z_i)$$

The measured distance d_i relates to the true distance by

$$r_i(\mathbf{p}) = \|\mathbf{p} - \mathbf{p}_i\| - d_i$$

where r_i is the residual of beacon i.

---

3. Variance and σ Incorporation

Each measurement has an associated variance s_i^2 that combines:

1. Range noise from RSSI
2. Beacon's advertised position uncertainty

3.1 Range variance (from RSSI noise)

Using first-order error propagation:

$$ \mathrm{Var}(d_i) \approx \left(\frac{\partial d_i}{\partial \mathrm{RSSI}_i} \right)^2 \mathrm{Var}(\mathrm{RSSI}_i) \;=\; \left(\frac{\ln(10)}{10\, n_i} \, d_i \right)^2 \sigma_{\mathrm{rssi},i}^2 $$

(All variables are scalar; this is KaTeX-safe.)

3.2 Beacon position variance

If a beacon advertises an isotropic uncertainty σ_i:

$\mathrm{Var}_{\mathrm{beacon},\, i} \approx \sigma_i^2$

3.3 Total variance per measurement

$s_i^2 = \mathrm{Var}(d_i) + \mathrm{Var}_{\mathrm{beacon},\, i}$

---

4. Weighted Least Squares Cost Function

The UGV's estimated position $\hat{\mathbf{p}}$ minimizes

$$ J(\mathbf{p}) = \sum_i \frac{r_i(\mathbf{p})^2}{s_i^2} = \sum_i \frac{ \left(\|\mathbf{p} - \mathbf{p}_i\| - d_i \right)^2 }{s_i^2} $$

This is equivalent to maximum likelihood estimation under Gaussian noise with heterogeneous variances.

---

5. Estimating Global σ

After solving for $\hat{\mathbf{p}}$, compute the overall uncertainty:

$$ \hat{\sigma} = \sqrt{ \frac{1}{N - 3} \sum_i r_i(\hat{\mathbf{p}})^2 } $$

where N is the number of beacons and 3 is the number of position parameters (x, y, z).

---

6. Robust Form (Optional)

To reduce the effect of outliers (for example, multipath), replace the quadratic cost with a robust loss such as Huber or Cauchy:

$$ J(\mathbf{p}) = \sum_i \rho\!\left(\frac{r_i(\mathbf{p})}{s_i} \right) $$

where $\rho(\cdot)$ is a smooth function that limits the influence of large residuals.

---

7. Summary

Symbol	Meaning
$\mathbf{p}$	Current UGV position (x, y, z)
$\mathbf{p}_i$	Beacon i position (x_i, y_i, z_i)
d_i	Distance inferred from RSSI
r_i	Residual = geometric distance – estimated distance
s_i	Combined standard deviation per beacon
σ_i	Advertised beacon position uncertainty
$\hat{\sigma}$	Estimated global uncertainty

The weighted least-squares estimator is:

$$ \hat{\mathbf{p}} = \arg\min_{\mathbf{p}} \sum_i \frac{ \left(\|\mathbf{p} - \mathbf{p}_i\| - d_i\right)^2 }{s_i^2} $$

bps: validations

- test -> introspect
- beacon -> receiver
- loop: 2 rpis, 3 rpis
- review
- data collection
- live: 1 rpi, 2 rpis, 2 rpis, anchor + 2 nodes

bps: validations: test-introspect

```
@bps install
```

```
@bps test
```

```
🪄  bluer_algo.bps.utils.test: connected to system bus with unique name:
:1.19
🪄  exported org.example.Hello at /org/example/Hello
🪄  run in another terminal: "@bps introspect unique_bus_name=:1.19"
```

in another terminal,

```
@bps introspect unique_bus_name=:1.19
```

```
s "Pong"
```

validate in the first window,

```
🪄  bluer_algo.bps.utils.test.ping() called by busctl!
```

tested on 2 rpis.

bps: validations: beacon-receiver

```
@bps beacon - - \
    --generate 1 \
    --sigma $(@random --float 1) \
    --x $(@random --float 1) \
    --y $(@random --float 1) \
    --z $(@random --float 1) \
    --timeout 60
```

on another rpi,

```
@bps receiver upload - \
    --grep sparrow+swallow \
    --timeout 10

history:
- hostname: sparrow3-back
  rssi: -52
  sigma: 88.34217834472656
  tx_power: -1.0
  x: 33.44688415527344
  y: 45.99796676635742
  z: 94.80066680908203
- hostname: sparrow3-back
  rssi: -51
  sigma: 88.34217834472656
  tx_power: -1.0
  x: 33.44688415527344
  y: 45.99796676635742
  z: 94.80066680908203
ping:
  hostname: sparrow2
  rssi: -1.0
  sigma: 1000.0
  tx_power: -1.0
  x: 0.0
  y: 0.0
  z: 0.0
```

bps: validations: loop-2

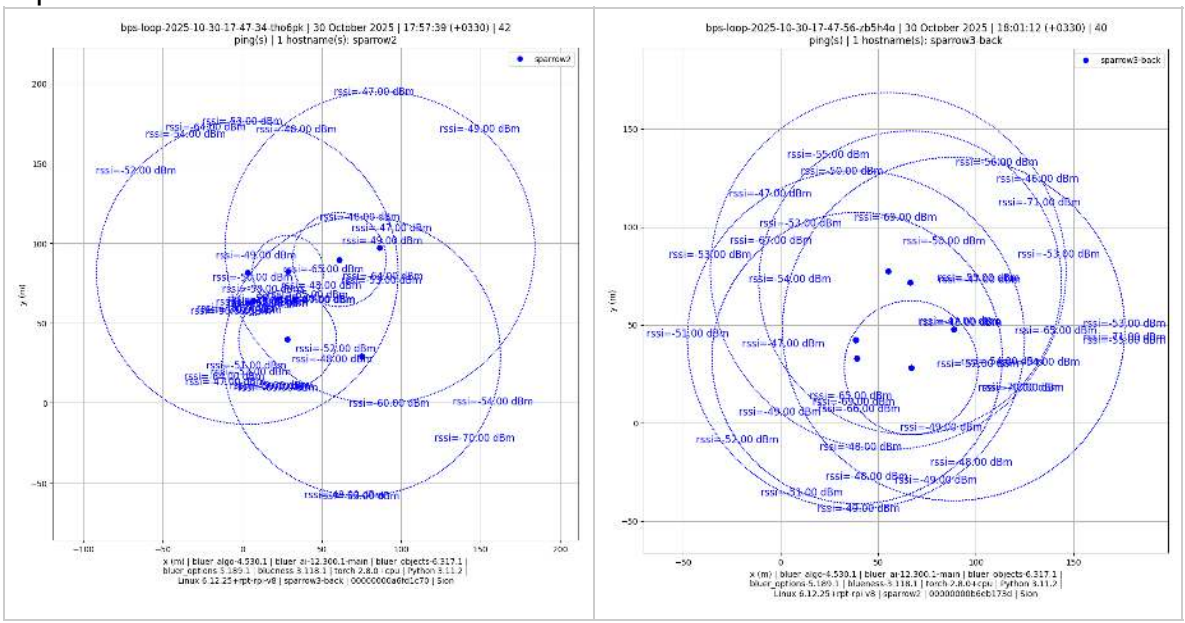
on 2 rpis,

```
@bps loop start simulate,upload
```

after a few minutes,

```
@bps loop stop [rpi <machine-name>]
```

▶ publication



▶ bps-loop-2025-10-30-17-47-34-tho6pk/metadata

▶ bps-loop-2025-10-30-17-47-56-zb5h4o/metadata

bps: validations: loop-3

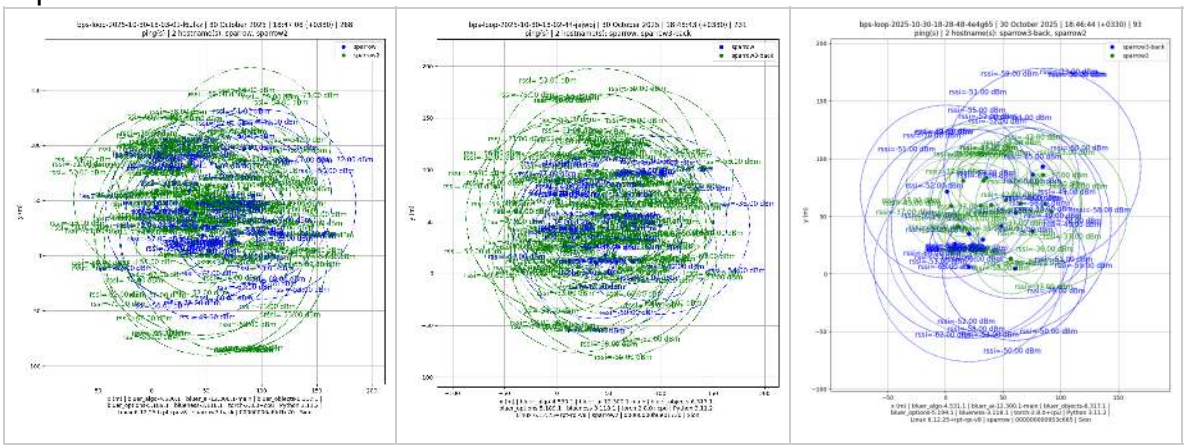
on 3 rpis,

```
@bps loop start simulate,upload
```

after a few minutes,

```
@bps loop stop [rpi <machine-name>]
```

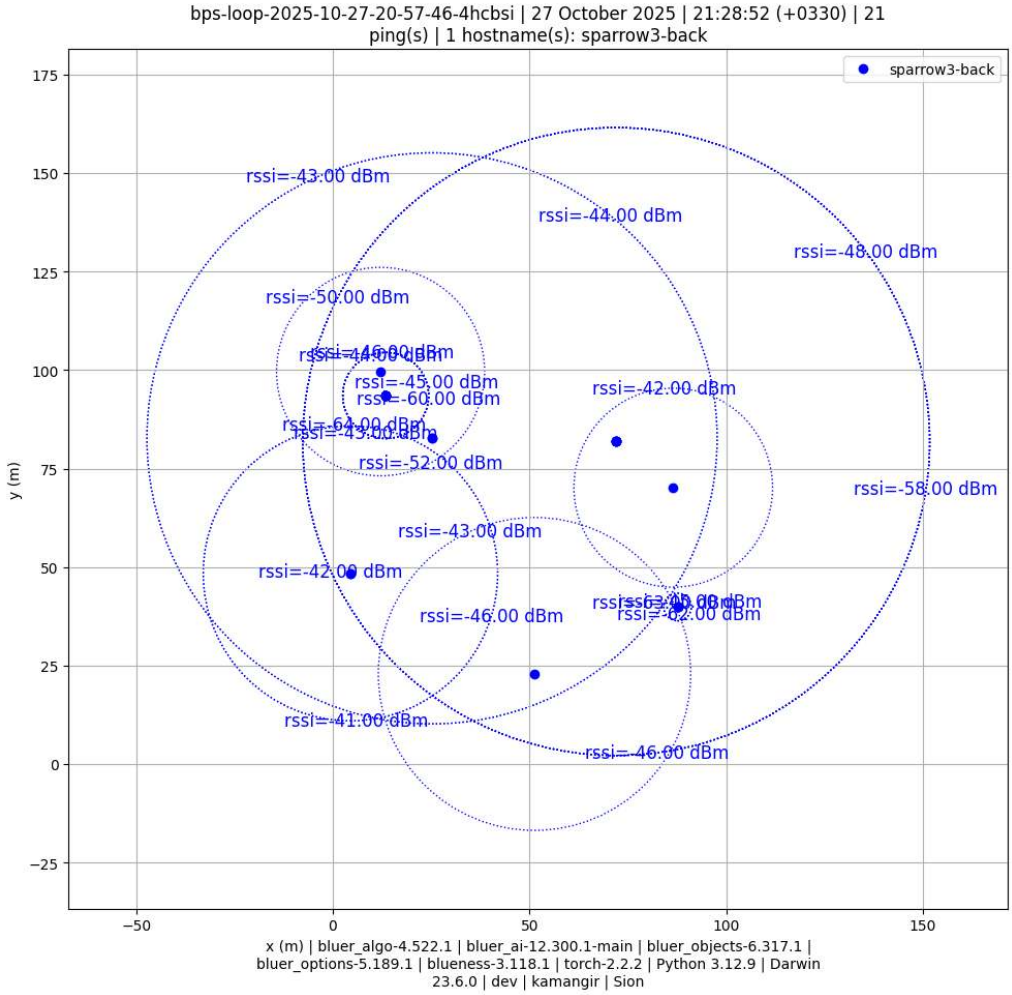
▶ publication



▶ bps-loop-2025-10-30-18-03-01-l6ulkz/metadata
▶ bps-loop-2025-10-30-18-02-44-jajwbj/metadata
▶ bps-loop-2025-10-30-18-28-48-4e4g65/metadata

bps: validations: review

```
@select bps-loop-2025-10-27-20-57-46-4hcbsi

@bps review upload .

@assets publish \
    extensions=png,push .
```



image

▶ metadata

bps: validations: data-collection

on 1 rpi:

```
@select bps-stream-$(@timestamp)
@bps generate - . \
    --sigma 1.0 \
    --x 0 \
    --y 0 \
    --z 0
@bps loop start upload .
```

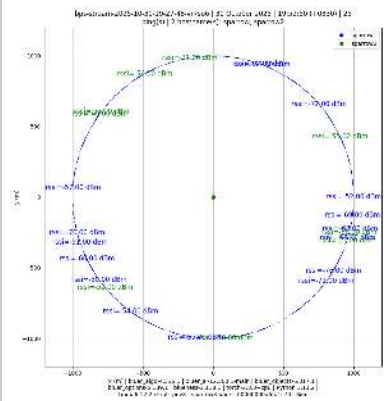
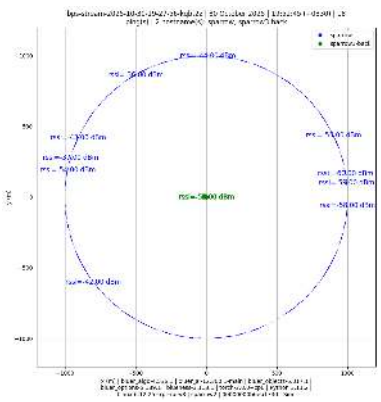
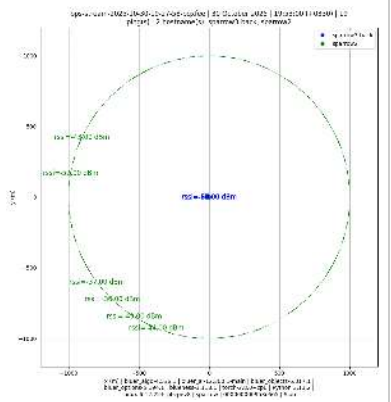
on 2 rpis,

```
@bps loop start upload
```

after a few minutes,

```
@bps loop stop [rpi <machine-name>]
```

▶ publication

bps-stream-2025-10-30-19-27-48-vr7so6	bps-stream-2025-10-30-19-27-56-kqbt2z	bps-stream-2025-10-30-19-27-58-pqxfee
		

▶ bps-stream-2025-10-30-19-27-48-vr7so6/metadata
▶ bps-stream-2025-10-30-19-27-56-kqbt2z/metadata
▶ bps-stream-2025-10-30-19-27-58-pqxfee/metadata

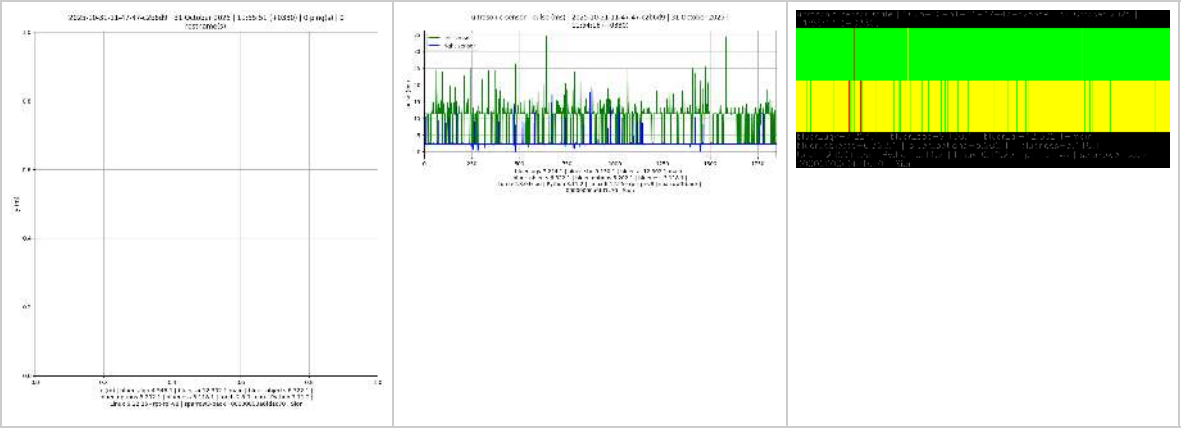
bps: validations: live-1

on one rpi,

```
@swallow env set bps 1

@env HAS_BPS

BLUER_SBC_SWALLOW_HAS_BPS=1

@select; @session start
```

▶ publication

2025-10-31-11-47-47-c2b6d9

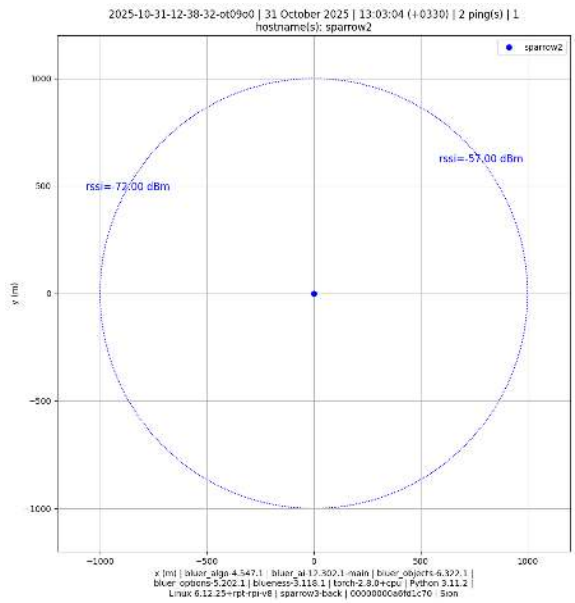
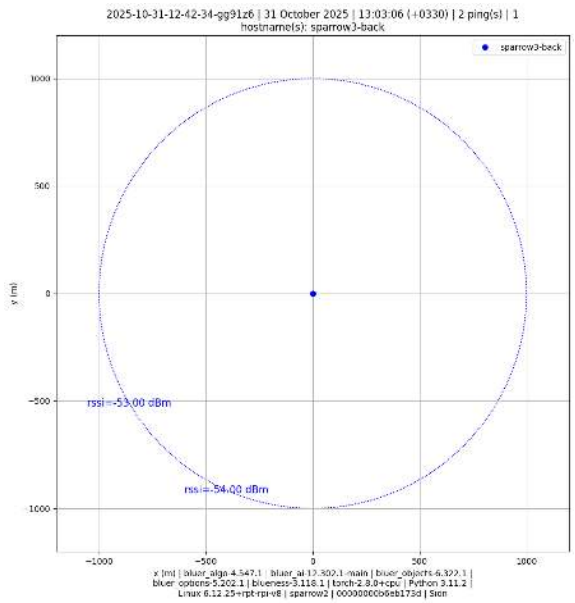
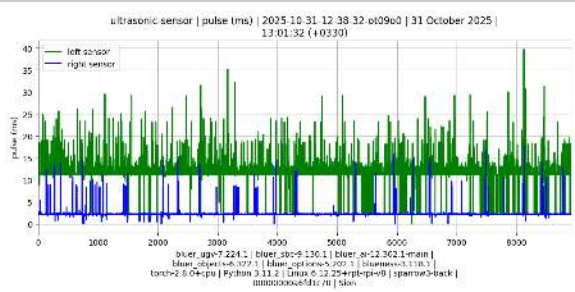
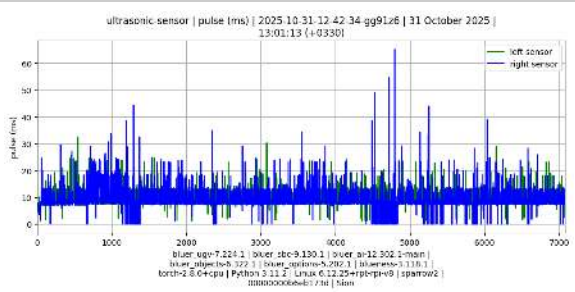
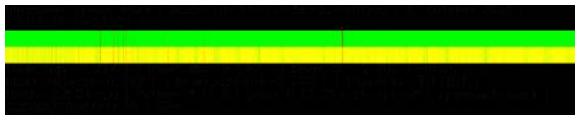
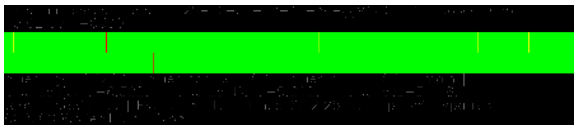


bps: validations: live-2

on 2 rpis,

```
@select; @session start
```

▶ publication

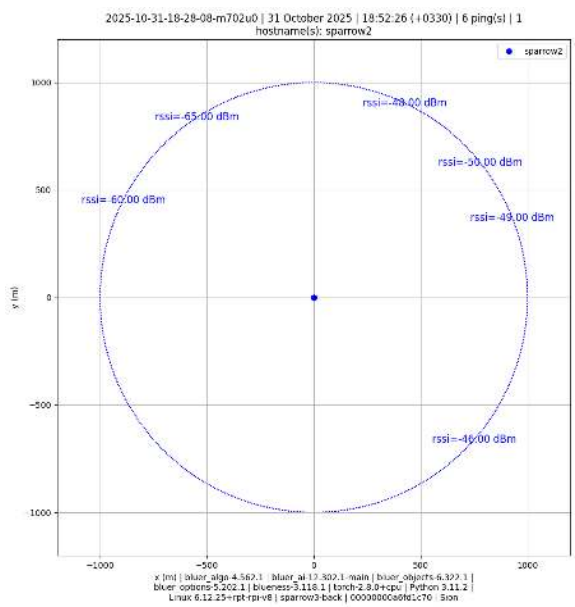
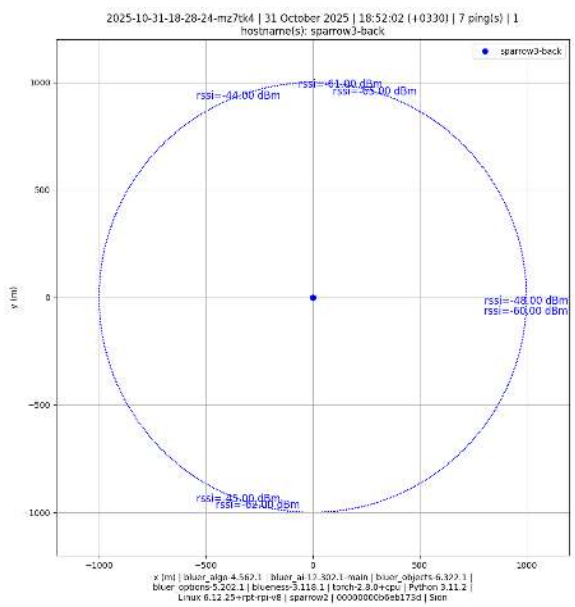
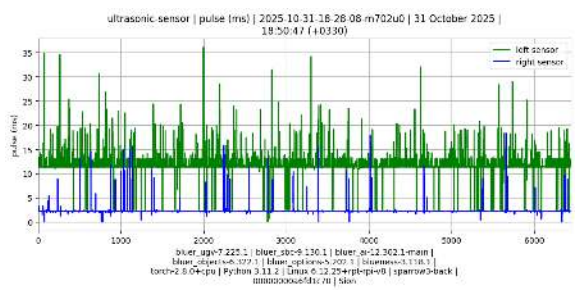
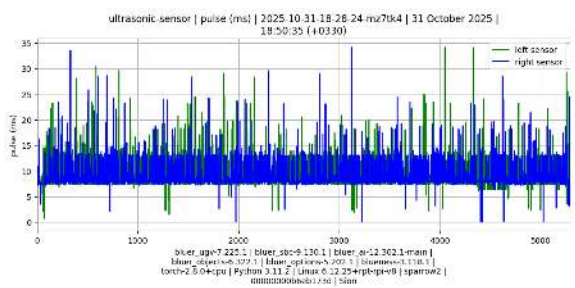
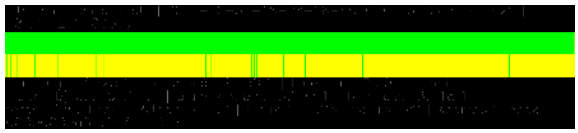
[2025-10-31-12-38-32-ot09o0](#)	[2025-10-31-12-42-34-gg91z6](#)
	
	
	

bps: validations: live-2b

on 2 rpis, continues live-2 after timing adjustments.

```
@select; @session start
```

▶ publication

2025-10-31-18-28-08-m702u0	2025-10-31-18-28-24-mz7tk4
	
	
	

bps: validations: live-3

on 3 rpis (1 anchor), continues live-2b with an anchor.

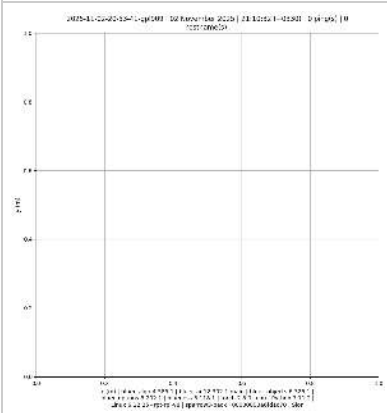
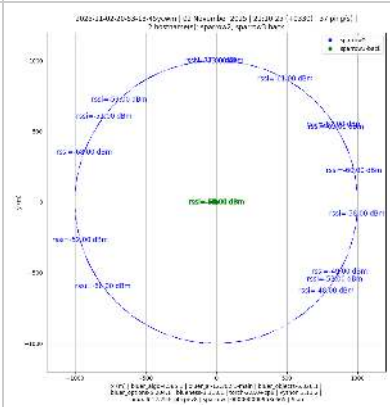
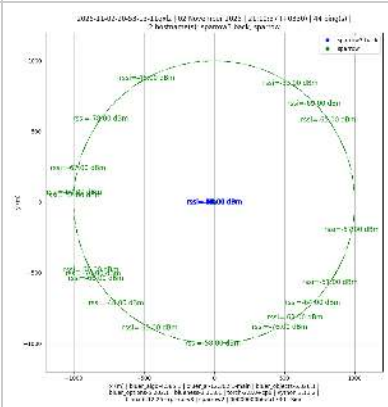
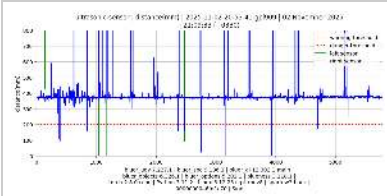
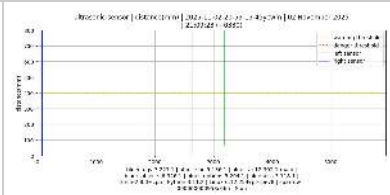
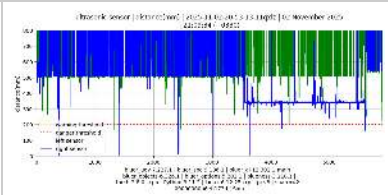
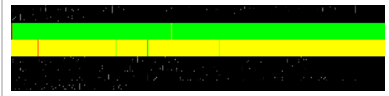
on the anchor run,

```
@bps set_anchor 0,0,0,1
```

on all run,

```
@select; @session start
```

▶ publication

2025-11-02-20-53-41-gpl909	2025-11-02-20-53-13-45ycwm	2025-11-02-20-53-13-11qxlz
		
		
		

bps: simulations: timing

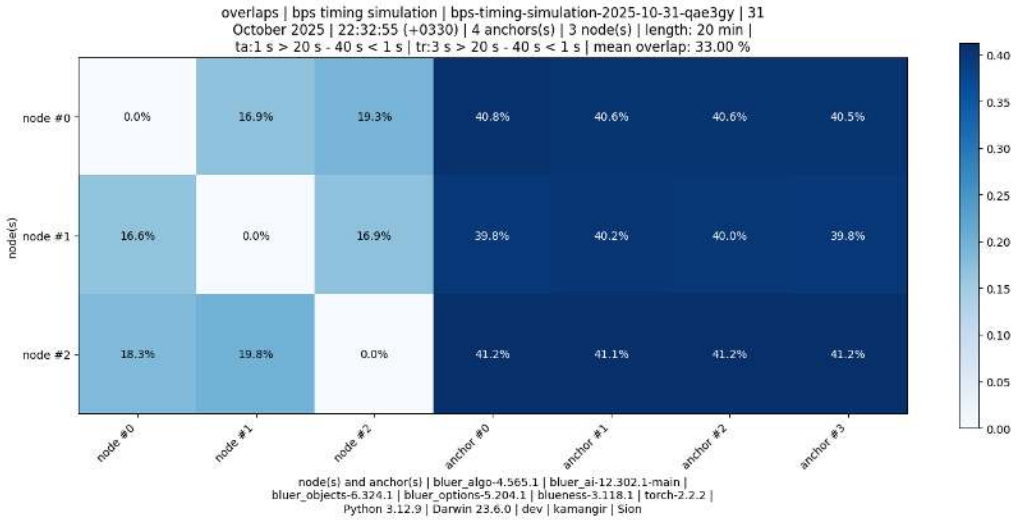
continues v1.

3 nodes

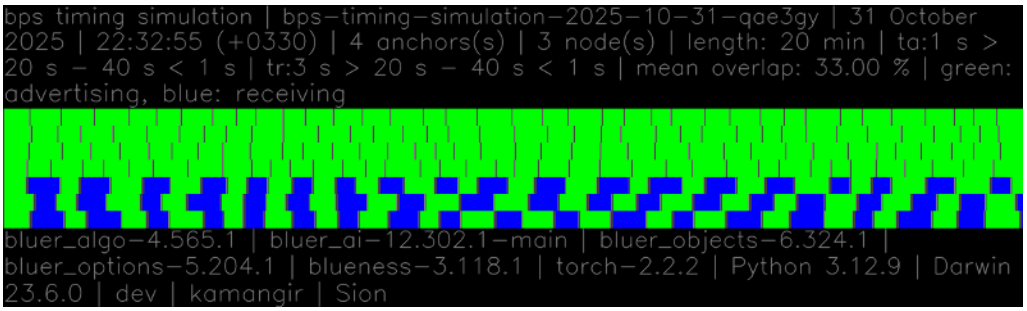
```
@select bps-timing-simulation-$(@@timestamp)

@bps simulate timing upload .

@assets publish extensions=png,push .
```

mean overlap: 0.33

▶ metadata



image



image

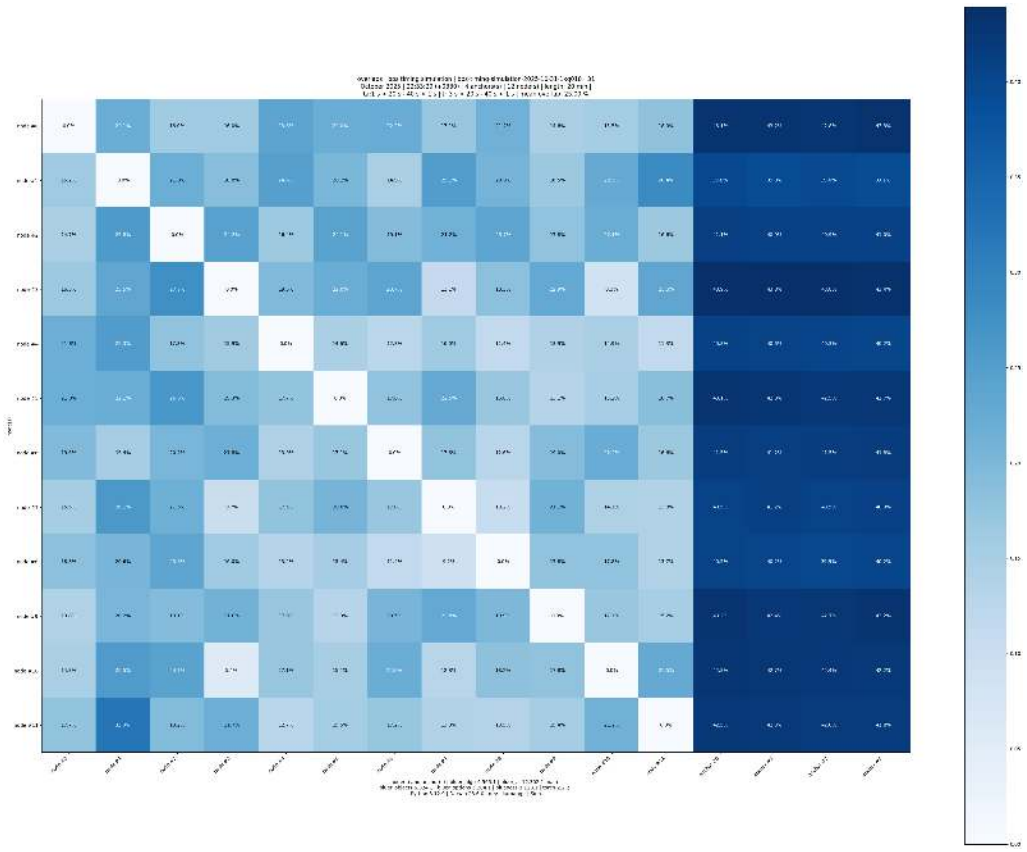
12 nodes

```
@select bps-timing-simulation-$(@@timestamp)

@bps simulate timing upload . \
    --nodes 12
```

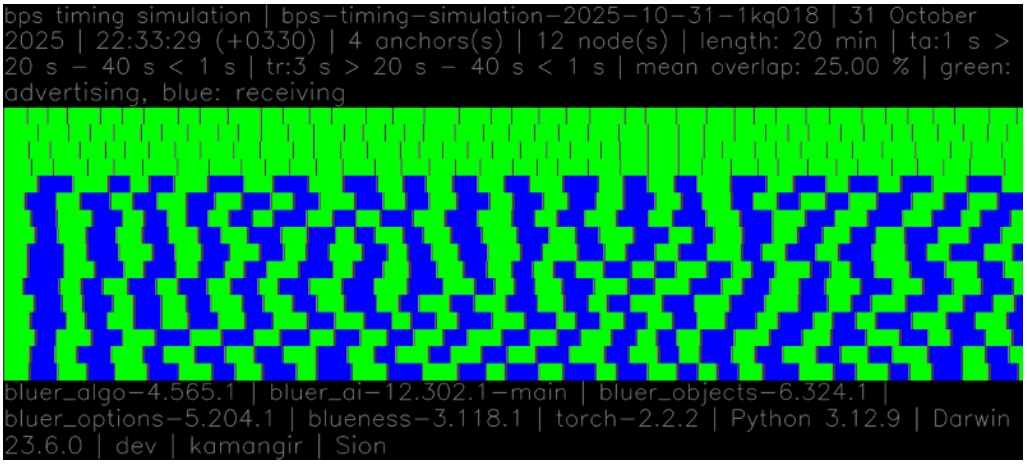
```
@assets publish extensions=png,push .
```

mean overlap: 0.25

▶ metadata



image



image

aliases: socket

```
@algo \
    socket \
    test \
    [dryrun] \
    [--host <host-name>] \
    [--what receiving | sending]
 . test socket.
```

socket

mac

both on mac.

```
@algo socket test - \
    --what receiving

@algo socket test - \
    --host dev.local \
    --what sending
```

✅

rpi -> mac

on mac:

```
@algo socket test - \
    --what receiving
```

on rpi:

```
@algo socket test - \
    --host dev.local \
    --what sending
```

✅

mac -> rpi

on rpi:

```
@algo socket test - \
    --what receiving
```

on mac:

```
@algo socket test - \
    --host swallow2.local \
    --what sending
```

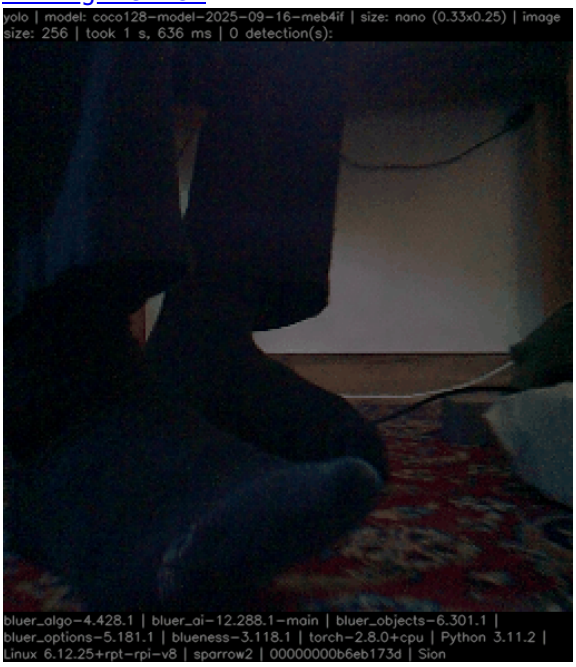
✅

validations

- timing-review
- village-1
- village-2
- village-3
- village-4
- village-5
- village-6
- village-7

village-1 	village-2 
village-3 	village-4 

timing-review



village-5

village-6

village-7

validations: timing-review

UGV(s): 🐬 arzhang2

single thread

```
::  in 1 min called 8 function(s):
::  #   0 - ClassicalUltrasonicSensor: called 388 time(s), total 46 s, avg
120 ms
::  #   1 - ClassicalYoloCamera: called 388 time(s), total 29 s, avg 076 ms
::  #   2 - ClassicalKeyboard: called 388 time(s), total 489 ms, avg 001 ms
::  #   3 - ClassicalLeds: called 388 time(s), total 081 ms, avg < 1 ms
::  #   4 - ClassicalRightMotor: called 388 time(s), total 072 ms, avg < 1
ms
::  #   5 - ClassicalLeftMotor: called 388 time(s), total 060 ms, avg < 1
ms
::  #   6 - ClassicalPushButton: called 388 time(s), total 025 ms, avg < 1
ms
::  #   7 - ClassicalSetPoint: called 388 time(s), total 005 ms, avg < 1 ms
```

loop frequency (Hz): 7

▶ yaml

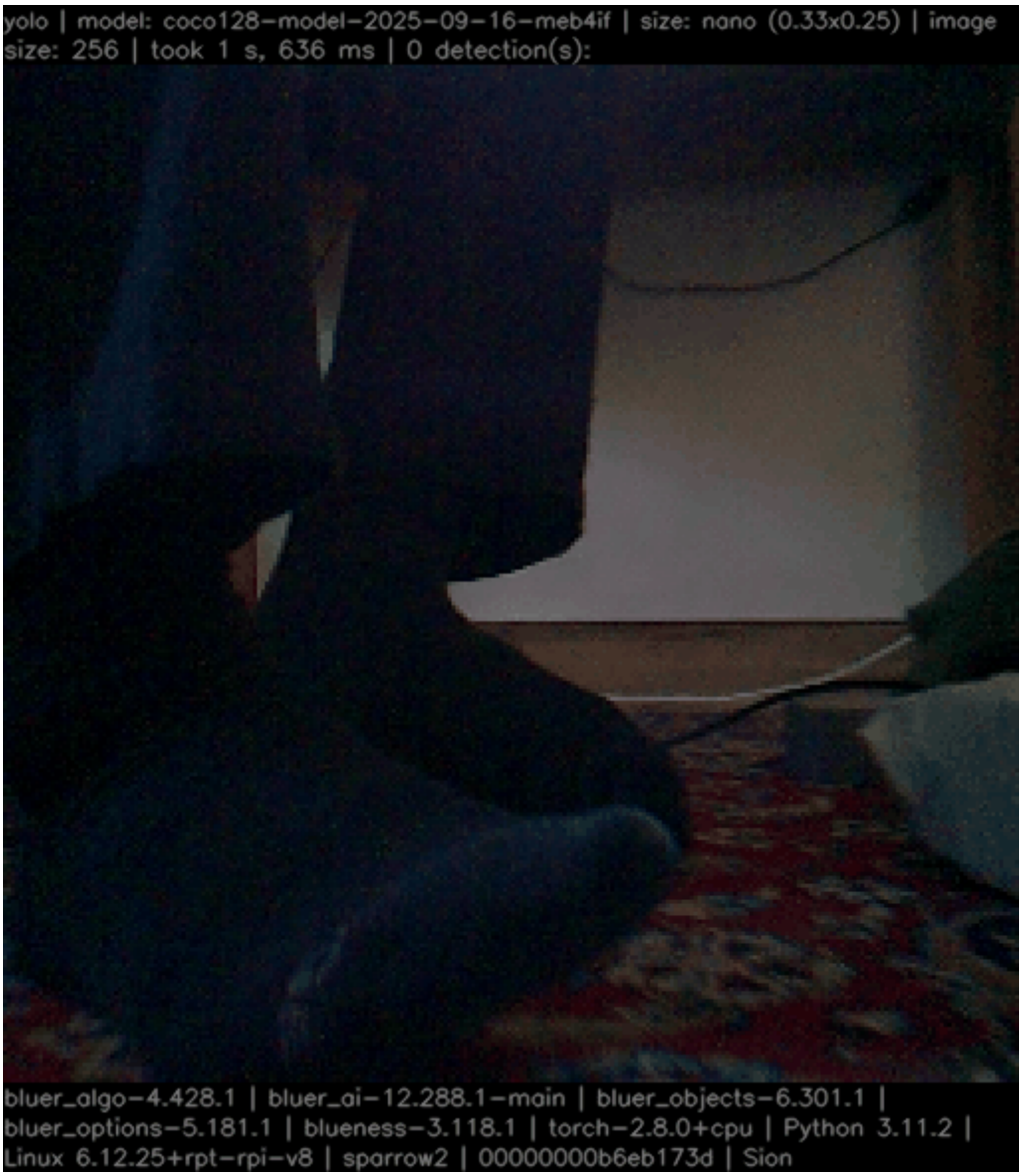
multi-threaded

yolo and ultrasonic run on individual threads.

```
::  in 5 min called 8 function(s):
::  #   0 - session.update: called 5,881 time(s), total 10 s, avg 002 ms
::  #   1 - ClassicalKeyboard: called 5,881 time(s), total 5 s, avg 001 ms
::  #   2 - ClassicalRightMotor: called 5,881 time(s), total 1 s, avg < 1
ms
::  #   3 - ClassicalLeds: called 5,881 time(s), total 1 s, avg < 1 ms
::  #   4 - ClassicalLeftMotor: called 5,881 time(s), total 888 ms, avg < 1
ms
::  #   5 - ClassicalPushButton: called 5,881 time(s), total 417 ms, avg <
1 ms
::  #   6 - ClassicalSetPoint: called 5,881 time(s), total 061 ms, avg < 1
ms
::  #   7 - ClassicalYoloCamera: called 5,881 time(s), total 033 ms, avg <
1 ms
```

loop frequency (Hz): 570.76

▶ yaml


yolo | model: coco128-model-2025-09-16-meb4if | size: nano (0.33x0.25) | image size: 256 | took 1 s, 636 ms | 0 detection(s):
bluer_algo-4.428.1 | bluer_ai-12.288.1-main | bluer_objects-6.301.1 | bluer_options-5.181.1 | blueness-3.118.1 | torch-2.8.0+cpu | Python 3.11.2 | Linux 6.12.25+rpt-rpi-v8 | sparrow2 | 00000000b6eb173d | Sion

image

validations: village-1

UGV(s): 🐬 arzhang

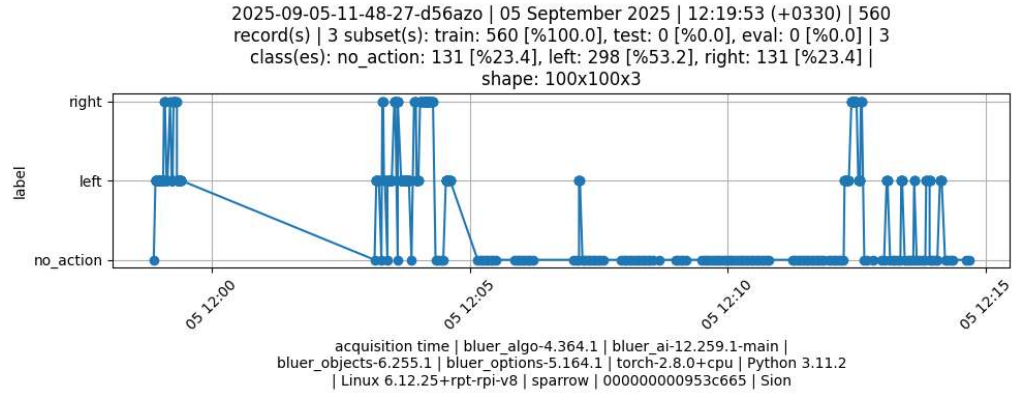
@select 2025-09-05-11-48-27-d56azo

@download policy=doesnt_exist

@upload public,zip

2025-09-05-11-48-27-d56azo

image



image

```
dataset:
  class_count: 3
  classes:
    0: no_action
    1: left
    2: right
  count: 560
  shape:
  - 100
  - 100
  - 3
  source: 000000000953c665
  subsets:
    eval: 0
    test: 0
    train: 560
```

image

observations

- image-classifier is validated. ✅

validations: village-2

UGV(s): 🐬 arzhang






debug objects

▶ collection

swallow-debug-2025-09-22-09-47-32-85hag3

swallow-debug-2025-09-22-09-59-29-emj29v

swallow-debug-2025-09-22-10-01-01-uzray6

swallow-debug-2025-09-22-10-06-19-hcyl1v

swallow-debug-2025-09-22-10-09-44-z6q9kn

swallow-debug-2025-09-22-10-19-35-mobajm



observations

- two wheel nuts loosened every few minutes. ⚠️
- one wheel nut tightened every few minutes. ⚠️

-> fixed in [village-3]

image

validations: village-3

UGV(s): 🐬 arzhang




debug object

▶ collection

swallow-debug-2025-09-25-13-16-59-rnm7jd

image

session object

```
dataset:
  count: 0
names:
  0: target
source: 000000000953c665
train: images/train
val: images/val
```

observations

- no wheel nuts loosened or tightened (: village-2) ✅
- bottom cover broke. -> 📜

validations: village-4

UGV(s): 🐬 arzhang2

debug object

```
runme() {
    @select $1

    @upload public,zip

    @assets publish extensions=gif,push
}

runme swallow-debug-2025-09-26-17-44-51-6pb87y
```

observations

- wheels functioned as expected. ✅
- robot rebooted after a minute of operation and again. ⚠️ loose power connections on shield found and fixed, subsequent testing validated the fix. ✅
- camera is upside down - fixed. ✅

```
runme swallow-debug-2025-09-27-19-15-31-6iq5vz
```

swallow-debug-2025-09-26-17-44-51-6pb87y	swallow-debug-2025-09-27-19-15-31-6iq5vz
	

validations: village-5

UGV(s): 🐬 arzhang, 🐬 arzhang2

script

```
runme() {
    @select $1

    @upload public,zip

    @assets publish extensions=gif,push

    # ---

    @select $2

    @swallow ultrasonic review \
        upload .

    @assets publish extensions=gif+png,push
}
```

objects

arzhang

```
runme \
    swallow-debug-2025-10-10-08-49-45-yk18ei \
    2025-10-10-08-52-35-6jjnzn
```

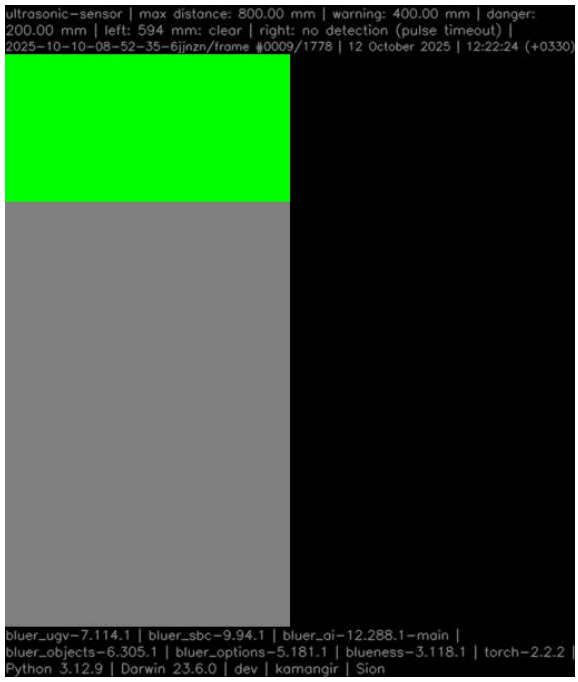
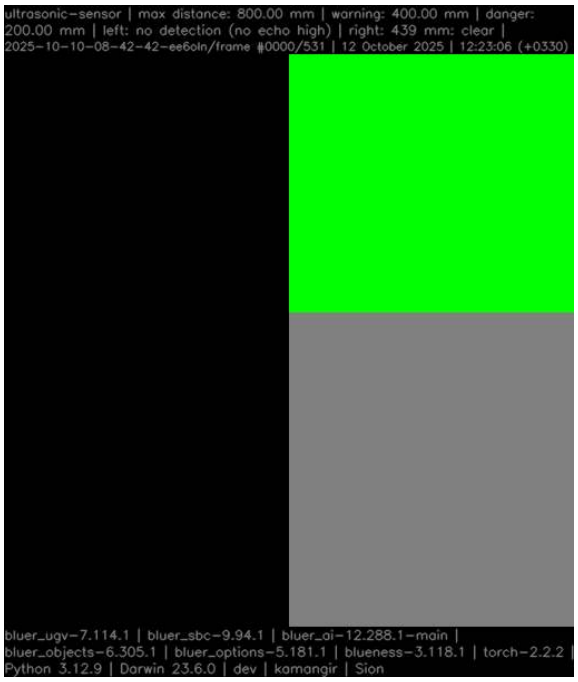
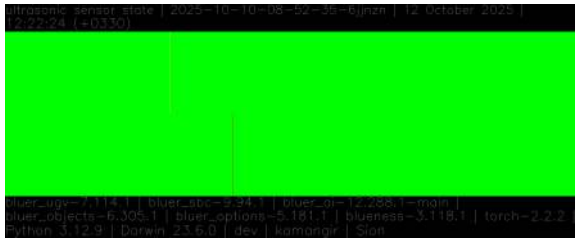
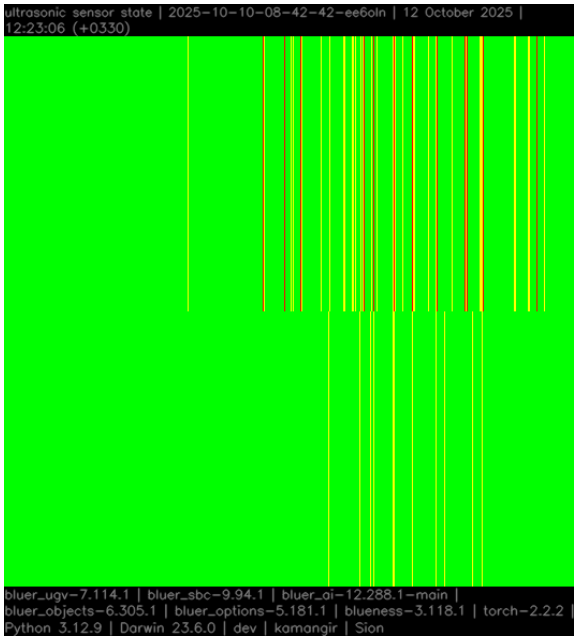
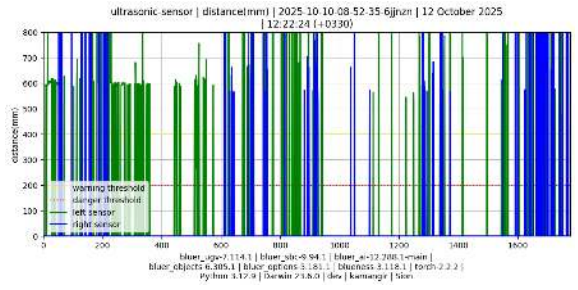
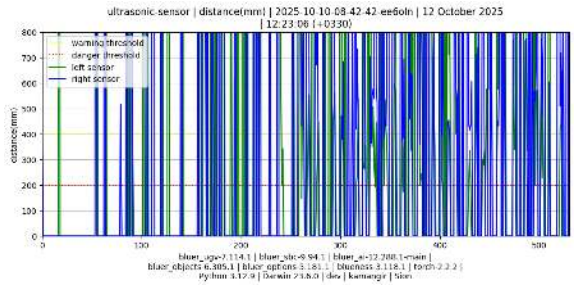
loop frequency (Hz): 310.87

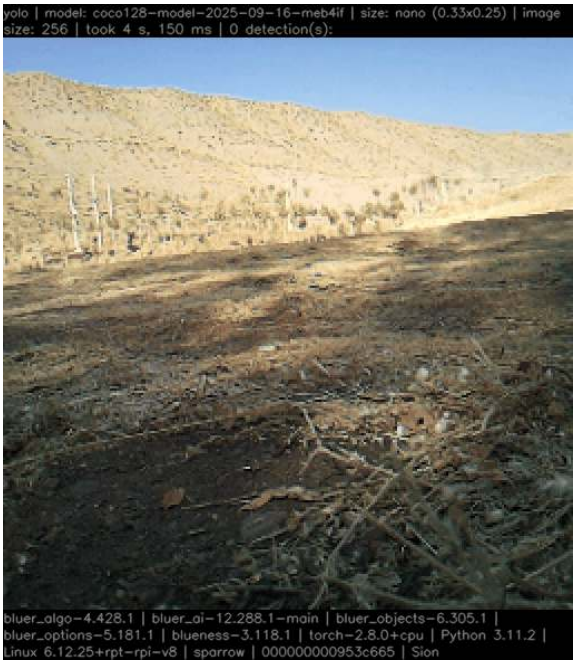
arzhang2

```
runme \
    swallow-debug-2025-10-10-08-40-38-k8oc2p \
    2025-10-10-08-42-42-ee6oln
```

loop frequency (Hz): 408.95

---

arzhang	arzhang2
	
	
	

arzhang	arzhang2
	

observations

- ultrasonic sensor is activated when the surface is uneven - will adjust the sensor.
- yolo may not have time to perform the action. -> 🪵

validations: village-6

UGV(s): 🐬 arzhang, 🐬 arzhang2

scripts

```
@select swallow-debug-2025-10-19-14-14-23-ectn97

@upload public,zip

@assets publish extensions=gif,push

runme() {
    @select $1

    @swallow ultrasonic review \
        upload .

    @assets publish extensions=gif+png,push
}

runme 2025-10-19-14-16-36-tunrlm
runme 2025-10-19-14-16-07-75yxbw
```

objects

arzhang

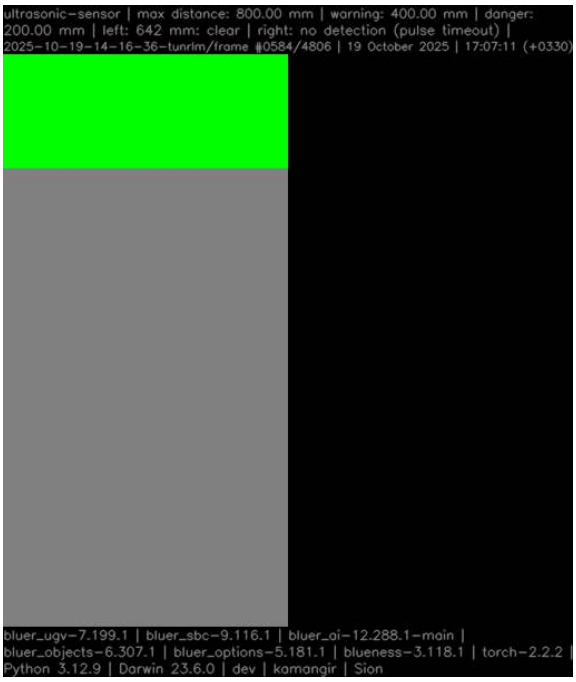
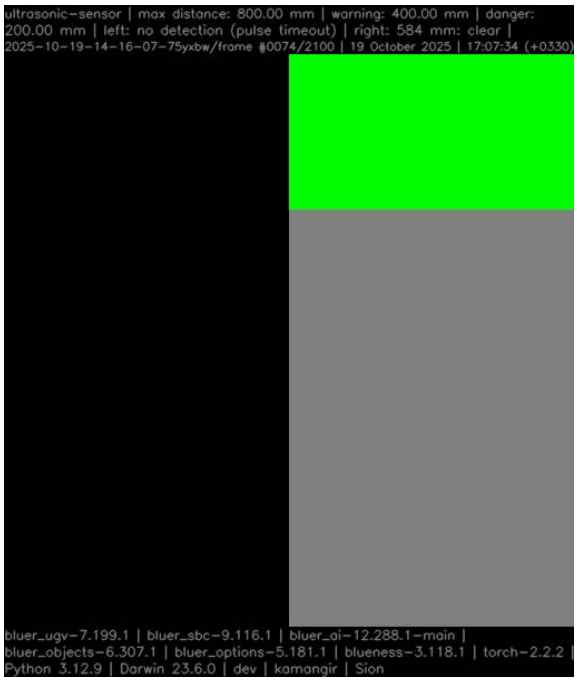
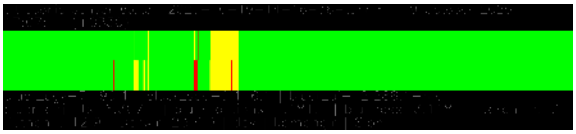
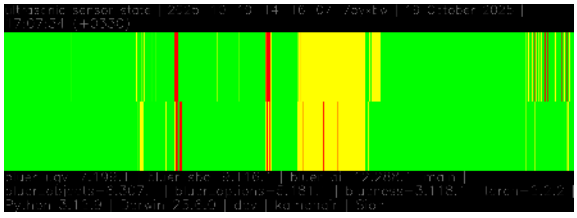
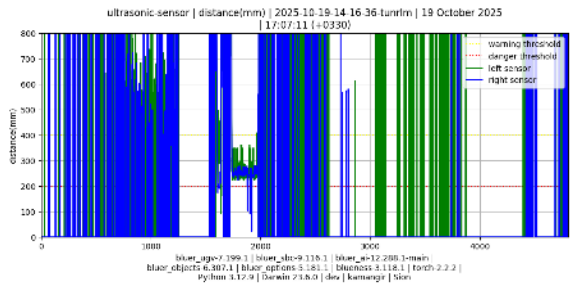
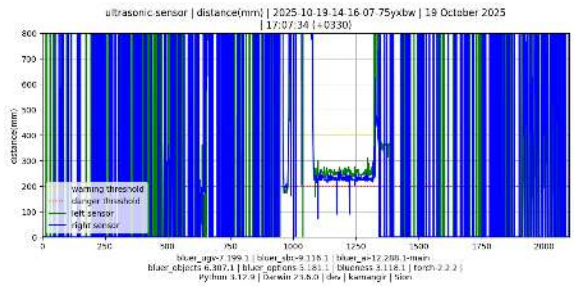
loop frequency (Hz): 321.78

arzhang2

loop frequency (Hz): 242.58

---

arzhang	arzhang2
2025-10-19-14-16-36-tunrlm	2025-10-19-14-16-07-75yxbw

[arzhang](#)	[arzhang2](#)
	
	
	

image

observations

- the range of numpad is ~10-20 m, noticeably lower than that of the full keyboard, which is ~50 m.



validations: village-7

UGV(s): 🐬 arzhang, 🐬 arzhang2, ⚓ arzhang3

▶ publication

arzhang

loop frequency (Hz): 185.88

2025-11-06-10-50-35-myadvn

arzhang2

loop frequency (Hz): 190.13

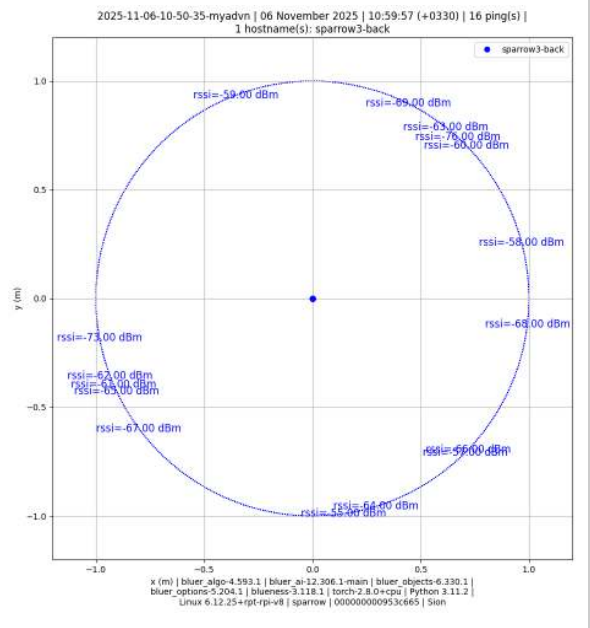
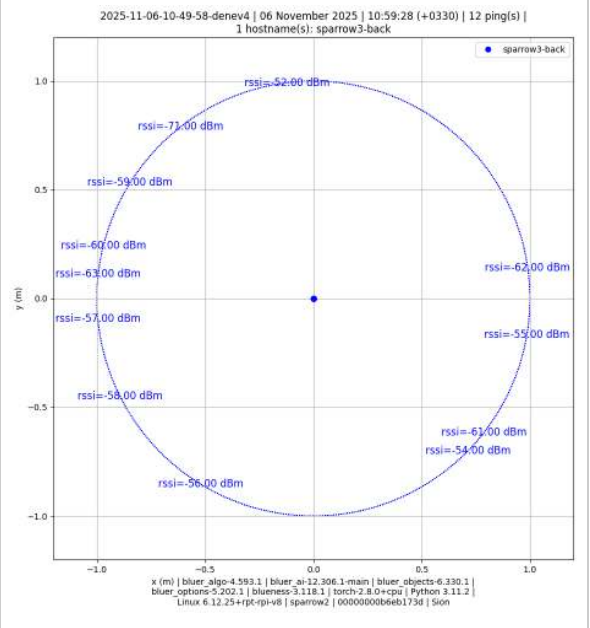
2025-11-06-10-49-58-denev4

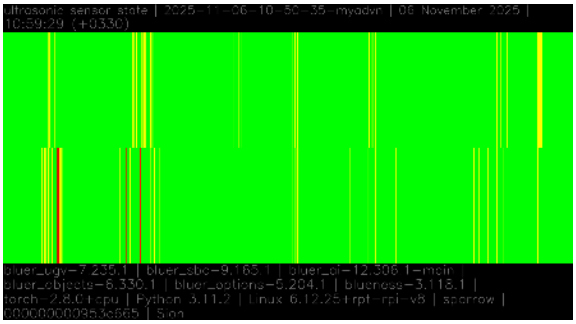
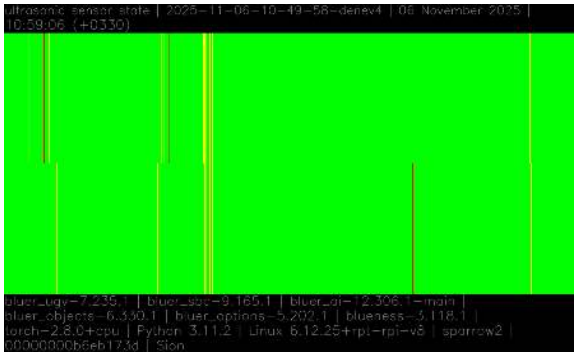
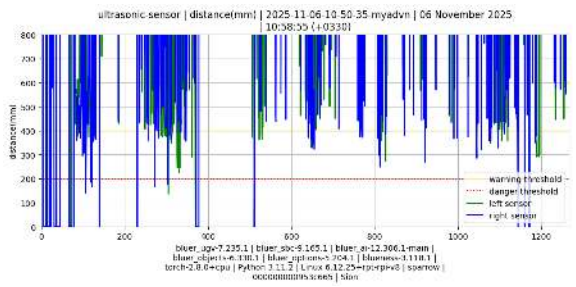
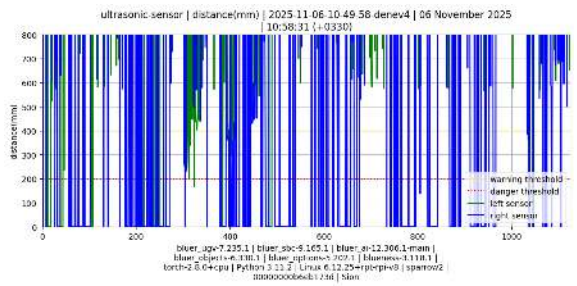
arzhang3

loop frequency (Hz): 183.64

2025-11-06-10-49-36-koxzf3

---

arzhang	arzhang2
	

arzhang	arzhang2
	
	

observations

- one of the back wheels on arzhang broke. -> 🪵
- the on/off switch on arzhang broke -> 🪵
- the two arzhangs did not receive each other's advertisements. they both received advertisements from arzhang3. ⚠️ - will review in the next validation.

bluer-objects

`bluer-objects` are the inputs and outputs of AI algo. They are maintained in cloud storage (supports WebDav) and their metadata is tracked by MLflow. Examples are the Sentinel-2 datacube `datacube-EarthSearch-sentinel_2_l1c-S2A_10UDC_20240731_0_L1C` and `@geo watch` outputs.

Also home to bluer README, and the `Lock`.

installation

```
pip install bluer-objects
```

aliases

@assets, @clone, @download, @gif, @host, @ls, @metadata, @mlflow, @pdf, @upload.

> `blue-objects` for the Global South.

built by `bluer README`, based on `bluer_objects-6.361.1`.

bluer-options

`bluer_options` implements an `options` argument for Bash.

installation

```
pip install bluer_options
```

if using outside the bluer-ai ecosystem, add this line to `~/.bash_profile` or `~/.bashrc`,

```
source $(python3 -m bluer_options locate)/.bash/bluer_options.sh
```

for more refer to 🔻 giza.

usage

let the function receive one or more `options` arguments (example below) then parse them with `bluer_ai_option`, `bluer_ai_option_int`, and `bluer_ai_option_choice`.

```
function func() {
    local options=$1

    local var=$(bluer_ai_option "$options" var default)
    local key=$(bluer_ai_option_int "$options" key 0)
    local choice=$(bluer_ai_option_choice "$options"
value_1,value_2,value_3 default)

    :
}
```

this enables the user to call `func` as,

```
func var=12,~key,value_1
```

all options have defaults and order doesn't matter.

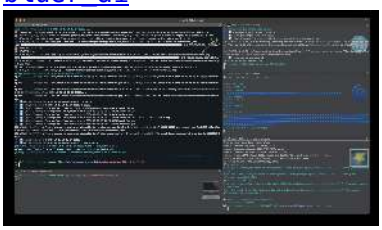
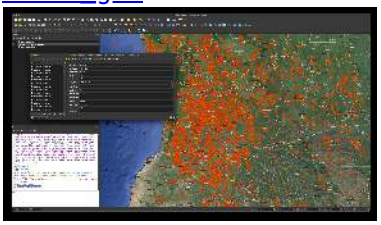
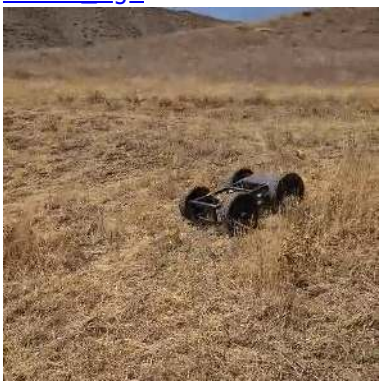
▶ example 1
▶ example 2

`blue-options` for the Global South.

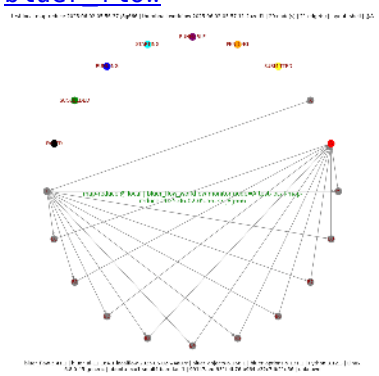
bluer-south

AI for Global South: Free from Big Tech.

```
pip install bluer-south
```

blog  kamangir writes here.	bluer_ai  A language to speak AI. pypi v12.332.1	bluer_sbc image AI for single board computers and related designs. pypi v9.246.1
bluer_geo  AI for a Blue Planet. pypi v5.83.1	bluer_ugv  AI x UGV. pypi v7.485.1	bluer_algo  AI Algo. pypi v4.602.1

[bluer_flow](#)



Workflow management.

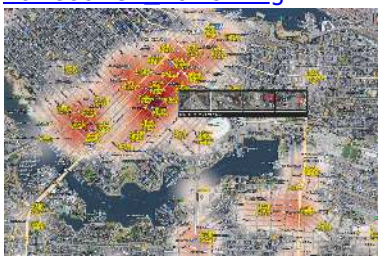
pypi v5.55.1

[bluer_academy](#)

an academy for practical AI in Iran.

pypi v5.179.1

[vancouver_watching](#)



Vancouver Watching with AI.

pypi v4.36.1

[bluer_journal](#)

A journal for the age of AI.

pypi v5.97.1

[bluer_options](#)

Options for Bash.

pypi v5.211.1

[bluer_objects](#)

Object management in Bash.

pypi v6.361.1

[bluer_plugin](#)

A git template for a bluer-ai plugin.

pypi v4.54.1

[bluer_sandbox](#)

A sandbox for ideas and experiments.

pypi v5.540.1

[blueness](#)

Blueness for Bluer AI.

pypi v3.118.1

gizai  Access, Automation, Analytics, AI - A Mathematical model for AI languages. pypi v7.338.1	abadpour  Arash Abadpour's CV. pypi v7.17.1	

---

pylint passing | pytest passing | bashtest passing | pypi v4.118.1 | downloads 0/day

built by bluer README, based on bluer_south-4.118.1.

bluer-sandbox

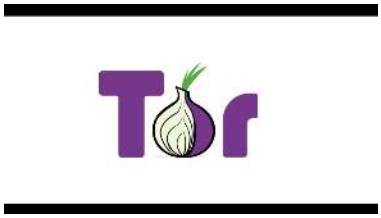
A sandbox for ideas and experiments.

installation

```
pip install bluer-sandbox
```

aliases

@arvancloud, @docker, @notebooks, @offline_llm, @speedtest, @tor, @v2ray. @village.

v2ray  tools to work with v2ray.	arvancloud  tools to work with arvancloud.	tor  tools to work with tor.
offline LLM  using llama.cpp.	bluer village  a bluer village.	

---

blue-sandbox for the Global South.

built by bluer README, based on bluer_sandbox-5.540.1.

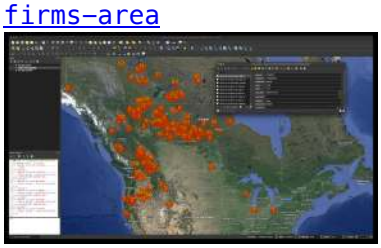
🌐 bluer-geo (@geo)

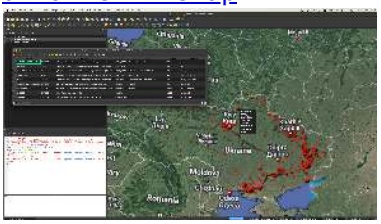
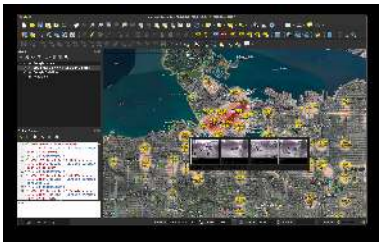
🌐 AI for a Blue Planet.

```
pip install bluer-geo
```

```
graph LR
    catalog_browse["@catalog<br>browse<br>&lt;catalog-name&gt;
<br>&lt;resource&gt;"]

    catalog_get["@catalog<br>get &lt;thing&gt;<br>--catalog
&lt;catalog&gt;"]

    catalog_list_catalogs["@catalog<br>list catalogs"]

    catalog_list["@catalog<br>list collections|datacube_classes<br>--
catalog &lt;catalog&gt;"]

    catalog_query["@catalog<br>query<br>&lt;catalog-name&gt;
<br>&lt;collection-name&gt; -<br>&lt;query-object-name&gt;"]

    catalog_query_and_ingest["@catalog<br>query<br>&lt;catalog-name&gt;
<br>&lt;collection-name&gt;<br>ingest,scope=&lt;scope&gt;<br>&lt;query-
object-name&gt;"]

    catalog_query_read["@catalog<br>query<br>read -<br>&lt;query-object-
name&gt;"]

    catalog_query_ingest["@catalog<br>query<br>ingest -<br>&lt;query-
object-name&gt;<br>scope=&lt;scope&gt;"]

    datacube_crop["@datacube<br>crop -<br>&lt;object-name&gt;
<br>&lt;datacube-id&gt;"]

    datacube_get["@datacube<br>get<br>catalog<br>&lt;datacube-id&gt;"]

    datacube_ingest["@datacube<br>ingest<br>scope=&lt;scope&gt;
<br>&lt;datacube-id&gt;"]

    datacube_label["@datacube<br>label -<br>&lt;datacube-id&gt;"]

    datacube_list["@datacube<br>list<br>&lt;datacube-id&gt;<br>--scope
&lt;scope&gt;"]

    geo_watch["@geo watch<br>batch<br>&lt;query-object-
name&gt;|target=&lt;target&gt; -<br>to=&lt;runner&gt; - -<br>&lt;object-
name&gt;"]

    catalog["🌐 catalog"]:::folder
    datacube_1["🧊 datacube"]:::folder
    datacube_2["🧊 datacube"]:::folder
    datacube_3["🧊 datacube"]:::folder
    terminal["💻 terminal"]:::folder
    QGIS["🖼️ QGIS"]:::folder
```

```
query_object["📂 query object"]:::folder
object["📂 object"]:::folder
target["🎯 target"]:::folder

catalog_list_catalogs --> terminal

catalog --> catalog_browse
catalog_browse --> terminal

catalog --> catalog_get
catalog_get --> terminal

catalog --> catalog_list
catalog_list --> terminal

catalog --> catalog_query
catalog_query --> query_object

catalog --> catalog_query_and_ingest
catalog_query_and_ingest --> query_object
catalog_query_and_ingest --> datacube_1

query_object --> catalog_query_read
catalog_query_read --> datacube_1

query_object --> catalog_query_ingest
catalog_query_ingest --> datacube_1
catalog_query_ingest --> datacube_2
catalog_query_ingest --> datacube_3

datacube_1 --> datacube_crop
target --> datacube_crop
datacube_crop --> datacube_1

datacube_1 --> datacube_get
datacube_get --> terminal

datacube_1 --> datacube_ingest
datacube_ingest --> datacube_1

datacube_1 --> datacube_list
datacube_list --> terminal

datacube_1 --> datacube_label
datacube_label --> QGIS
datacube_label --> datacube_1

query_object --> geo_watch
target --> geo_watch
geo_watch --> object

classDef folder fill:#999,stroke:#333,stroke-width:2px;
```

Maxar_Open_Data	copernicus	firms-area
		
catalog: Maxar's Open Data program	catalog: Copernicus Data	catalog: Fire Information for Resource

	Space Ecosystem - Europe's eyes on Earth	Management System (FIRMS).
ukraine-timemap  catalog: Bellingcat Civilian Harm in Ukraine TimeMap dataset, available through this UI and this API.	QGIS  An AI terraform for QGIS.	global-power-plant-database  The Global Power Plant Database is a comprehensive, open source database of power plants around the world datasets.wri.org.
geo-watch  Watch the planet's story unfold.	catalog  Generalized STAC Catalogs.	datacube  Generalized STAC Items.

---

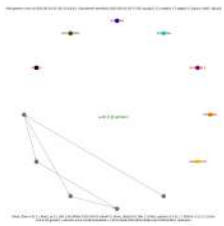
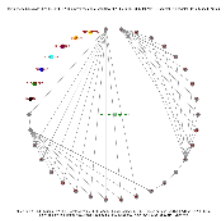
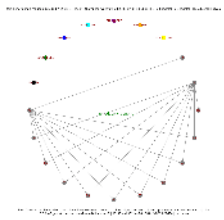
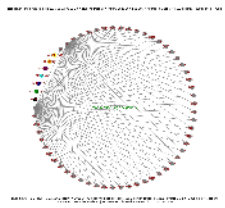
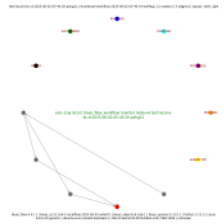
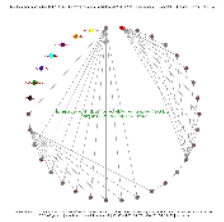
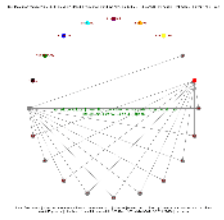
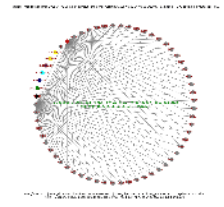
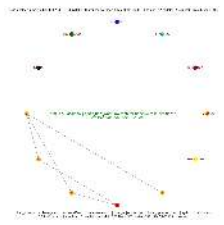
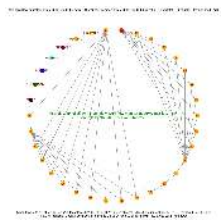
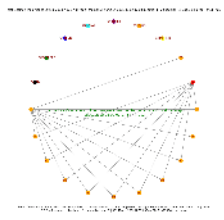
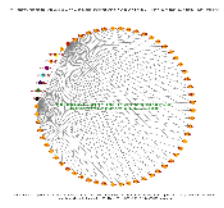
blue-geo for the Global South.

built by bluer README, based on bluer_geo-5.83.1.

📜 bluer_flow

📜 `bluer_flow` for workflow management.

```
pip install bluer_flow
```

📜	a-bc-d	hourglass	map-reduce	map-reduce-large
generic	 🔗	 🔗	 🔗	 🔗
local	 🔗	 🔗	 🔗	 🔗
localflow	 🔗	 🔗	 🔗	 🔗

💡 example use: literature review using OpenAI API.

aliases

localflow, workflow.

> 🌀 `blueflow` for the Global South.

---

built by 🌀 bluer README, based on 📜 bluer_flow-5.55.1.

🌀 bluer-plugin

🌀 `@plugin` is a git template for a `bluer-ai` plugin, to build things like these, that out-of-the-box support,

- a github repo with actions.
- pylint.
- pytest.
- a pip-installable python + bash package published to pypi.
- a bash command interface.
- bash testing.
- in-repo compiled READMEs. example: template.md -> README.md.
- object management with cloud persistence with metadata tracking by MLflow.

installation

```
pip install bluer-plugin
```

creating a bluer-plugin

1️⃣ create a new repository from this template,

2️⃣ complete `<repo-name>` and `<plugin-name>` and run,

```
@git clone <repo-name> cd

@plugins transform <repo-name>

# review and clean up the repo.

pip3 install -e .

@init

@help @<plugin-name>
```

features

feature_1	feature_2	feature_3
		
description of feature 1 …	description of feature 2 …	description of feature 3 …

aliases

@plugin.

---

blue-plugin for the Global South.

---

pylint passing | pytest passing | bashtest passing | pypi v4.54.1 | downloads 0/day

built by bluer README, based on bluer_plugin-4.54.1.

🔻 giza

🔻 `giza` is a Mathematical model for AI languages such as `bluer-ai` and its ecosystem. This model is described in the working paper “Access, Automation, Analytics, AI” [tex].

Access, Automation, Analytics, AI

Arash Abadpour - arash@abadpour.com

April 15, 2025

"... the four A's that we're after ... (1) Accessibility when I ask a question I want to be able to access the data that allows me to answer the question that I'm asking (2) Automation our ability to make routine tasks that are presently done by humans so that they can be done by machines ... (3) Analytics we want to be able to generate insights that might not otherwise be obvious to us (4) AI ... - *James C. Slife*, The Future of Warfare: Preparing U.S. Military Forces for Competition and Contestation, GSF 2024 [1]."

Abstract

First, we develop a mathematical model to discuss the "Four A's" in Section 1. Then, we propose expansions for Access and Automation in Sections 2 and 3, respectively. We briefly review a proposed view of Analytics as Access to the outputs of Automation in Section 4. Finally, in Section 5, we review a reference implementation of the proposed framework [2] based on *Bash* [3] expansions that call into *Python* [4] in an AI application.

Contents

1 Theoretical Framework

Hypergraph of Objects & Commands

A group of *operators* maintain a growing space of *commands* in a set of *repositories*, using a system such as *git* [5] and following a collective peer-reviewed *pull-request* [6] process. Each operator can access a set of *machines* on a system such as *SageMaker* [7] and create *shells* on them to run commands to produce *objects* which have an implicit or explicit value.

The objective of every operator is to increase the quality and quantity of the objects they generate. Therefore, the operators are interested in mechanisms that enable them to build the commands in ways that allow them to generate large quantities of objects that adhere to requirements that are known in the future from objects that may not exist yet. This is particularly important because commands run other commands and some objects are intermediaries in generating other objects.

As such, we model an AI system as an infinite *hypergraph* [8] where the nodes are the objects and the edges are the commands. We note that this is a Hypergraph, and not a regular graph, because any edge can modify or create zero or more objects from zero or more existing objects. One subset of commands take in zero objects and generate zero objects. Instead, these commands modify the *state* of one or more machines or shells, and thus participate in the generation of the objects further down the line.

Whereas, a conventional hypergraph generalizes by allowing the edges to connect any two subsets of nodes, here, we are also interested in a second generalization that allows an edge to "call" other edges. An important example of this process is when an operator builds an algo (an edge) and then

submits a command to *AWS Batch* [9] that calls the algo 10,000 times on separate subsets of objects. Similarly, an operator may call a command that deploys the algo as an API that is called on a stream of incoming objects.

Hypergraphs have been used in the past to model parallel data structures [10], for data mining [11], and clustering [12]. For a list of other relevant uses of hypergraphs refer to [8, 13]. Moreover, the focus of this work is the development of mathematical tools for the optimal combination of algorithms as black boxes towards practical objectives. This is different from optimizing the internal workings of the same algorithms [14, 15].

State Machines

A machine is a state machine that is connected to many other machines and shares some of its state with them for read and write. A shell is a stateful access mechanism to a machine that an operator uses to run commands. In a 2022 survey of developers, 89% responded that they have a terminal open at least half of the day [16]. Running a command in a shell can potentially modify the state of all other machines. Two examples of machines are a Raspberry Pi [17] that runs Linux and is connected to the AWS infrastructure [18] and a docker container [19] running in AWS Batch. GNU Bash [3] is an example of a shell. See [20] for a comparison of multiple shells.

The operators act asynchronously while communicating with each other. Multiple operators may simultaneously use the same machine, and the same operator may simultaneously use multiple machines. Only one operator uses a shell at one time. Some machines are exogenous to this model, yet the operators can access their states in read or write modes through running commands. Cloud storage [21] and compute resources [9] are examples of these machines.

We, therefore, recognize the existence of a plurality of interconnected state machines. Objects are artifacts on some of these machines, and, therefore, their content is a component of the state of the machine(s) that carry them. Hence, the hypergraph is a subset of the state of the universal state machine.

Commands & Expansions

A command is a string of characters that is meaningful to *Bash* [3]. Bash is a "Unix shell and command language first released in 1989 that has been used as the default login shell for most Linux distributions" [22]. A shell is a "macro processor that executes commands" [23], where "macro processor means functionality where text and symbols are expanded to create larger expressions" [23]. There are seven kinds of *expansions* [24] in Bash.

Expansions

Brace Expansion [25] expands '`a{d,c,b}e`' to '`ade ace abe`'. *Tilde Expansion* [26] relates to words that begin with an unquoted tilde character (~). *Parameter and Variable Expansion* [27] enable the use of variables, as `${variable}`, as well as more elaborate pattern matching forms such as `${parameter/#pattern/string}`. *Command Substitution* "allows the output of a command to replace the command itself" [28]. *Arithmetic expansion* [29] enables arithmetic operations using the form `$(( expression ))` and *Word Splitting* [30] governs the splitting of the command to words. Finally, *Filename Expansion* [31] enables the familiar wildcard reference to filenames using '*' and '?'.

Command Syntax

We are interested in a special category of valid bash commands [32] that start with a specially–crafted callable, continue with a prescribed sequence of identifiers, and end with arguments. A *callable* is a valid command with no space and control operators [33]. Some of the well-known callables are *git* [5], *docker* [19], *pushd* [34], and *nano* [35].

Callables

In this convention, the type of each identifier is known based on the command until that identifier, and is one of the following.

- A `value`, either numerical or a filename, for example.
- An *options*.
- An object name or pointer.

This is an example command,

```
vanwatch ingest \
    area=vancouver,~batch,count=5,gif . \
    --count 12
```

Here, "`vanwatch`" is the callable and "`ingest`" is the task, effectively an Enum, which is an `Options`. The command continues with "`area=vancouver, batch,count=5,gif`", which is an `Options`, ".", which is an object pointer, and "`--count 12`", which are the arguments.

Conventions

argparse [36], *click* [37], *fire* [38], and many other command line parsers support the `--<keyword> <value>` convention. These arguments are captured in commands by ending calls to Python with

`--<keyword> <value>`

"$@:<number>". Command substitution [28] is useful where a delimited list of keywords is generated and consumed, such as in `for` loops. We recommend controlling these lists using the three arguments `--count`, `--delim`, and `--offset`.

`--delim space`

```
local object_name
for object_name in $(<command-1> \
    --count 2 \
    --offset 1 \
    --delim space); do
    <command-2> $object_name <args>
    [[ $? -ne 0 ]] && return 1
done
```

`@env`

System state is carried in a set of environment variables that are generaly `export`ed during initialization and are used to harmonize paths and other machine–specific parameters. These variables are listed by `@env [<keyword>]`. `@help <command>` shows help about `<command>`. `@init <options>` initialize the core, and therefore all plugins, and `<plugin-name> init <options>` initializes `<plugin-name>`.

`@help` `@init`

2 Access

Callable Expansion

When the callable `<func>` receives a command that starts with the task `<task>`, it calls the function `<func>_<task>` with the rest of the command, if such function exists. For example, here, the callable `bluer_ai_conda`, which is aliased to `@conda`, is defined.

```
#! /usr/bin/env bash

function bluer_ai_conda() {
    local task=$1

    local function_name=bluer_ai_conda_$task
    if [[ $(type -t $function_name) == "function" ]]; then
        $function_name "${@:2}"
        return
    fi

    conda "$@"
}

bluer_ai_source_caller_suffix_path /conda
```

This mechanism works with the function `bluer_ai_conda_exists`, which resides in `/conda/exists.sh`, listed below,

```
#! /usr/bin/env bash

function bluer_ai_conda_exists() {
    local options=$1
    local environment_name=$(bluer_ai_option "$options" name bluer_ai)

    if conda info --envs | grep -q "^$environment_name "; then
        echo 1
    else
        echo 0
    fi
}
```

The result is the *super-command* `@conda` which behaves as `conda`, except when the function `bluer_ai_conda_<task>` is defined. Here is an example of the outcome assembled together,

```
[[ $(@conda exists name=<env-name>) == 1 ]] && echo "found."
```

We use this *namespacing* [39] mechanism for organization as well as orchestration. In practice, this expansion yields callables with multiple prefixes. We generate `@<keyword>` aliases [40] to facilitate user access, as stated above. In practice, the use of this expansion is comparable to fitting the spanning tree of an Application Programming Interface (API) [41] to a dataset of use of digital tools by operators recorded as sequences of keystrokes and mouse clicks.

Options Expansion

An options is a string representation of a dictionary, such as,

```
<keyword-1>=<value-1>,<keyword-2>=<value-2>,...,<keyword-3>,-<keyword-4>},...
```

Options is implemented using basic Python and, therefore, the *options expansion* is available to Bash commands through command substitution [28]. In practice, the two additional expansions `@option::int` and `@option::choice` cast the output to an integer and select it from a list (equivalent to an Enum [42]), respectively.

```
value=$(@option "$options" <keyword> [<default>])

value=$(@option::int "$options" <keyword> 0 | 1)

value=$(@option::choice "$options" <comma,separated,list> <default>)
```

Object Expansion

Commands manipulate data as objects. An object is a uniquely named collection of files and folders and can be downloaded or uploaded, in part or as a whole. Some objects already exist in the environment. For example, an *item* in a *STAC collection* [43] (a datacube) or a dataset in *Kaggle* [44] are objects. A curated dataset, a model trained on it, and the model's predictions on a datacube, are examples of other objects.

Object Pointers

An object may be *selected*,

```
@select <object-name>
```

When `<object-name>` is selected, '`.`' expands to `<object-name>`. Similarly, '`..`', '`...`', and so on, as deep as needed, expand to the names of the previously selected object and the one before that. Commands default the objects they consume and modify to '`.`', '`..`', and so on. Therefore, because the commands in a script generally use the same objects, selecting the objects enables their names to be replaced with pointers. Often the defaults of the commands are designed to enable the omission of the pointers as well.

Object Metadata

Every object carries a `metadata.yaml` file that can be `get` and `post` as a dictionary.

```
@metadata get key=<key> [.|<object-name>]

@metadata post <key> <value> - [.|<object-name>]
```

Object Tags

An object can have many tags. A tag is key-value pair that is maintained in a system such as *MLflow* [45] and is `set` and `get`, and can be `search`ed,

```
@mlflow tags get [.|<object-name>] [--tag <tag>]

@mlflow tags search [<keyword-1>=<value-1>,<keyword-2>,~<keyword-3>]

@mlflow tags set [.|<object-name>] [<keyword-1>=<value>,<keyword-2>,~<keyword-3>]
```

Object Persistence

Object are persisted on AWS S3 [21]. Objects can be downloaded, uploaded, and listed,

```
@download [filename=<filename>] [.|<object-name>]

@upload [filename=<filename>] [.|<object-name>]

@list cloud|local [.|<object-name>]
```

3 Automation

Prefixing

A `<prefix>` can run a `<command>` with certain `<options>` through the *prefixing expansion*,

```
<prefix> <options> <command>
```

This expansion enables submitting a command to another machine, including a *docker* [19] container, and compute resources, such as *AWS Batch* [9] and Argo Workflows [46].

```
@eval [dryrun,path=<path>] <command>

@docker eval [dryrun] <command>

@batch eval [dryrun,name=<job-name>] <command>
```

`workflows`

A `workflow` is a set of commands that depend on each other because they consume objects that are generated by other commands, among other reasons. We model a workflow as a Directed Acyclic Graph (DAG), where each node is a command. We use this abstraction through `NetworkX` [47] to run workflows on AWS Batch [9] and Argo Workflows [46]. In practice, there is often a series of commands that generate a cascade of objects that contains the final products. Each command in this series has one or more options and takes in and produces zero or more objects. Here is a representative case,

Cascading Options

```
vanwatch \
    ingest \
    target=<target>,count=<count> \
    $object_name

vanwatch \
    detect \
    count=<count>,gif,model=<model-id>,publish \
    $object_name \
    --overwrite 1
```

The *options cascade expansion* compresses this cascade of commands as follows,

```
vanwatch \
    ingest \
    target=<target>,count=<count> \
    $object_name \
    detect,count=<count>,gif,model=<model-id>,publish \
    --overwrite 1
```

List of Objects

A series of objects that are tagged by a unique tag to represent a list of objects. Or, the list of objects may be maintained in the metadata of an object. This object or tag expands to the list of objects, through tag or metadata mechanisms.

Terraforming

Terraforming is the process of installing modules and adjusting the configuration of a shell or a machine for a specific purpose through running a series of commands. These commands are often long and unintuitive and contain secrets and other environment variables. Terraforming is generally persistent and has to be done once at the first use of a machine. Examples include `ssh` into a new machine, starting a machine on a service such as AWS SageMaker [7], and running commands inside the Python Console in *QGIS* [48].

`@seed`

The *seed expansion* generates the code required for terraforming a target and transfers it through one of the listed mechanisms,

```
@seed [<target>] [clipboard|filename=<filename>|usb-key|scp|screen]
```

4 Analytics

We understand Analytics as Access to the outcomes of Automation. See Sections 2 and 3.

5 AI

`vanwatch` [49] discovers traffic cameras in a target, ingests images from them, and runs detection algo on the images to generate time series analytics.

`discover`

Cameras are discovered through public websites and, therefore, they are represented in different formats in different targets. `vanwatch discover` discovers the cameras in `<target>` and stores them in `<target>.geojson` in the object `<object-name>` and tags the object for discovery by `ingest`.

```
vanwatch \
    discover \
    [target=<target>,count=<-1>,dryrun,~tag,~upload] \
    [-|<object-name>]
```

Here, `target=<target>,count=<-1>,dryrun, tag, upload` is an options. If the object pointer `-` is selected, a timestamped object prefixed `<target>-discover` is generated. This options input also enables `dryrun`, a limited count, and the ability to disable tagging and uploading the object. These features are handy for testing and special runs.

`vanwatch discover` points to the function `vancouver_watching_discover`[1] through callable expansion and the alias `vanwatch`.

```
#! /usr/bin/env bash

function vancouver_watching_discover() {
    local options=$1
    local target=$(bluer_ai_option "$options" target vancouver)
    local do_dryrun=$(bluer_ai_option_int "$options" dryrun 0)
    local do_tag=$(bluer_ai_option_int "$options" tag $(bluer_ai_not $do_dryrun))
    local do_upload=$(bluer_ai_option_int "$options" upload $(bluer_ai_not $do_dryrun))
```

`options` uses options expansion.

```
    local object_name=$(bluer_ai_clarify_object $2 \
        $target-discover-$(bluer_ai_string_timestamp_short))
```

`object_name` uses object expansion.

```
    local function_name=vancouver_watching_discover_$target
    if [[ $(type -t $function_name) != "function" ]]; then
        bluer_ai_log_error "vancouver_watching: discover: $target: target not found."
        return 1
    fi
```

A callable expansion.

```
    bluer_ai_clone \
        ~relate,~tags \
        $VANWATCH_QGIS_TEMPLATE \
        $object_name

    bluer_ai_log "discovering $target -> $object_name"
    bluer_ai_eval ,$options \
        $function_name \
        ,$options \
        $ABCLI_OBJECT_ROOT/$object_name \
        "${@:3}"
    local status="$?"

    [[ "$do_upload" == 1 ]] &&
        bluer_objects_upload - $object_name

    [[ "$status" -ne 0 ]] && return $status

    [[ "$do_tag" == 1 ]] &&
        bluer_objects_mlflow_tags set \
```

[1] https://github.com/kamangir/vancouver-watching/blob/main/vancouver_watching/.abcli/discover.sh

```
            $object_name \
            app=vancouver_watching,target=$target,stage=discovery

    return $status
}

bluer_ai_source_caller_suffix_path /discovery
```

The callable expansion above points to the function `vancouver_watching_discover_vancouver`[2].

```
#! /usr/bin/env bash

function vancouver_watching_discover_vancouver() {
    local options=$1
    local object_path=$2

    curl \
        https://opendata.vancouver.ca/explore/dataset/web-cam-url-links/download/?format=geojson \
        >$object_path/detections.geojson

    local count=$(bluer_ai_option_int "$options" count -1)

    bluer_ai_eval ,$options \
        python3 -m vancouver_watching.discover \
        discover_cameras_vancouver_style \
        --filename $object_path/detections.geojson \
        --prefix https://trafficcams.vancouver.ca/ \
        --count $count \
        "${@:3}"
}
```

`vanwatch ingest` finds the latest set of cameras discovered in `<target>` through tag search and ingests `count` images into `<object-name>` and then runs `vanwatch detect` unless `detect` is present. `object-name` is then tagged for future discovery.

```
vanwatch \
    ingest \
    [target=<target>,count=<-1>,~download,dryrun,~upload] \
    [-|<object-name>] \
    [detect,count=<-1>,~download,dryrun,gif,model=<model-id>,publish,~upload] \
    [--overwrite 1] \
    [--verbose 1]
```

`vanwatch ingest` generates an object from another that it finds through tag search. Note that this command uses arguments as well as two cascading options. The first controls the ingest of the images and the second is passed to `detect`.

`vanwatch detect` runs object detection algo [50] on the images ingested into `<object-name>` and updates `<target>.geojson`.

```
vanwatch \
    detect \
    [count=<-1>,~download,dryrun,gif,model=<model-id>,publish,~upload] \
    [.|<object-name>] \
    [--overwrite 1] \
    [--verbose 1]
```

[2] https://github.com/kamangir/vancouver-watching/blob/main/vancouver_watching/.abcli/discovery/vancouver.sh

References

[1] James C. Slife. The Future of Warfare - Preparing U.S. Military Forces for Competition and Contestation - GSF 2024 - YouTube. Center for Strategic & International Studies, YouTube, https://www.youtube.com/watch?v=cLmcqy5vJv4&t=2161s.

[2] Arash Abadpour. bluer-ai: a language to speak AI. GitHub repository, https://github.com/kamangir/bluer-ai.

[3] Bash - GNU Project - Free Software Foundation. https://www.gnu.org/software/bash/.

[4] Welcome to Python.org. https://www.python.org/.

[5] Git - git Documentation. https://git-scm.com/docs/git.

[6] About pull requests - GitHub Docs. https://docs.github.com/en/pull-requests/collaborating-with-pull-re

[7] Machine Learning Service - Amazon SageMaker - AWS. https://aws.amazon.com/sagemaker/.

[8] Alain Bretto. *Hypergraph Theory; An Introduction*. Springer, 2013.

[9] Efficient Batch Computing - AWS Batch - AWS. https://aws.amazon.com/batch/.

[10] B. Hendrickson and T.G. Kolda. Graph partitioning models for parallel computing. *Parallel Computation*, 26:1519–1545, 2000.

[11] C. Hébert, A. Bretto, and B. Crémilleux. A data mining formalization to improve hypergraph minimal transversal computation. *Fundamenta Informaticae*, 80(4):415–433, 2007.

[12] S.R. Bulò and M. Pelillo. A game-theoretic approach to hypergraph clustering. *proceedings of the NIPS*, pages 1571–1579, 2009.

[13] Brendan Fong and David I Spivak. Seven sketches in compositionality: An invitation to applied category theory, 2018.

[14] Gitta Kutyniok. The mathematics of artificial intelligence, 2022.

[15] Gabriel Peyré. The mathematics of artificial intelligence, 2025.

[16] Will McGugan. Why I Founded Textualize. https://www.textualize.io/blog/why-i-founded-textualize/, 2023. Accessed: 2025-01-01.

[17] Teach, learn, and make with the Raspberry Pi Foundation. https://www.raspberrypi.org/.

[18] Cloud Computing Services - Amazon Web Services (AWS). awshttps://aws.amazon.com/.

[19] Use the Docker command line - Docker Docs. https://docs.docker.com/engine/reference/commandline/cli/.

[20] Michael Greenberg and Austin J. Blatt. Executable formal semantics for the posix shell, 2019.

[21] Amazon S3 - Cloud Object Storage - AWS. https://aws.amazon.com/s3/.

[22] Bash (unix shell) - wikipedia. https://en.wikipedia.org/wiki/Bash_(Unix_shell).

[23] What is a shell? (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/What-is-a-

[24] Shell Expansions (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Shell-Expa

[25] Brace Expansion (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Brace-Expa

[26] Tilde Expansion (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Tilde-Expa

[27] Shell Parameter Expansion (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Sh

[28] Command Substitution (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Comma

[29] Arithmetic Expansion (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Arithm

[30] Word Splitting (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Word-Splitt

[31] Bash Startup Files (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Filename

[32] Shell Syntax (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Shell-Syntax.

[33] Definitions (Bash Reference Manual): control operator. https://www.gnu.org/software/bash/manual/html_node/

[34] Directory Stack Builtins (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Dire

[35] nano - Text editor. https://www.nano-editor.org/.

[36] argparse — Parser for command-line options, arguments and sub-commands — Python 3.12.4 documentation. https://docs.python.org/3/library/argparse.html.

[37] Click - The Pallets Projects. https://palletsprojects.com/p/click/.

[38] Python Fire. https://google.github.io/python-fire/.

[39] Namespace - Wikipedia. https://en.wikipedia.org/wiki/Namespace#Emulating_namespaces.

[40] Aliases (Bash Reference Manual). https://www.gnu.org/software/bash/manual/html_node/Aliases.html.

[41] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine, 2000.

[42] enum — Support for enumerations — Python 3.12.4 documentation. https://docs.python.org/3/library/enum.html.

[43] SpatioTemporal Asset Catalog (STAC) Specification. Introduction to STAC. Accessed: 2024-12-22.

[44] Qi Chen, Lei Wang, Yifan Wu, Guangming Wu, Zhiling Guo, and Steven L Waslander. Aerial imagery for roof segmentation: A large-scale dataset towards automatic mapping of buildings. *ISPRS Journal of Photogrammetry and Remote Sensing*, 147:42–55, 2019.

[45] MLflow. https://mlflow.org/.

[46] Argo Workflows - Kubernetes-native workflow engine supporting DAG and step-based workflows. https://argoproj.github.io/workflows/.

[47] NetworkX Developers. NetworkX. https://networkx.org/, 2023.

[48] 4. Features — QGIS Documentation documentation. https://docs.qgis.org/3.34/en/docs/user_manual/preambl

[49] Arash Abadpour. Vancouver Watching: Vancouver watching with AI. GitHub repository, https://github.com/kamangir/Vancouver-Watching.

[50] Ultralytics HUB - YOLOv8x. https://hub.ultralytics.com/models/R6nMlK6kQjSsQ76MPqQM?tab=preview.

built by `gizai-7.338.1`.