طرح درس Chovgan: جلسه‌به‌جلسه، از اولین حرکت تا تیم کامل

هر جلسه حدود ۹۰ دقیقه است و دقیقاً یکی از درس‌های کد پایه (عدد LESSON در `chovgan_team/robot.py`) را دنبال می‌کند. همه‌ی کدها با Python نوشته شده‌اند و در خود شبیه‌ساز CHOVGAN 1.0.0 اجرا و آزمایش شده‌اند. U14 می‌تواند جلسه‌های ۱ تا ۸ را آرام‌تر و با تمرین بیشتر جلو ببرد؛ U19 همه‌ی جلسه‌ها را کار می‌کند. آموزش خط‌به‌خط هر درس در صفحه‌ی «آموزش از صفر» و در فایل `TUTORIAL-FA.md` داخل کد پایه است.

جلسه‌ی ۱: آشنایی: نصب، منوها و اولین مسابقه بدون کد

هدف

شبیه‌ساز را نصب و اجرا کنند، کد پایه را بارگذاری کنند و یک مسابقه‌ی کامل را از شروع تا نتیجه ببینند.

مفاهیم

- `CHOVGAN-1.0.0-windows-x86_64.zip` را باز کنید و `CHOVGAN.exe` را اجرا کنید. Python جدا لازم نیست.
- منو: Settings، Rules & Guide، Robot Lab، Training، Competition.
- زمین ۱۰۵ در ۶۸ متر است. x در طول زمین از 52.5- تا 52.5+ و y در عرض از 34- تا 34+. دروازه‌ها روی x = ±52.5.
- هر تیم چهار ربات دارد: B1 تا B4 (آبی) و Y1 تا Y4 (زرد). هر ربات کد خودش را جداگانه اجرا می‌کند.
- شبیه‌ساز گل، خطا، اوت و شروع مجدد را خودش ثبت می‌کند. نیمه‌ی دوم جای تیم‌ها عوض می‌شود.
- بدون کد VALID، مسابقه و تمرین شروع نمی‌شوند و ربات‌ها خودبه‌خود حرکت نمی‌کنند.

فعالیت

- `Chovgan-BaseCode.zip` را دانلود کنید. Competition → رده → نام تیم‌ها → در STRATEGY همین ZIP را برای هر دو تیم بدهید → READY.
- در طول بازی Esc بزنید: Controller status را باز کنید و ببینید هر هشت ربات RUNNING هستند.
- با کلیدهای ۱ تا ۷ نماهای دوربین را عوض کنید. Tactical (کلید ۶) کل زمین را از بالا نشان می‌دهد.

ارزیابی

آبی در نیمه‌ی اول به کدام دروازه حمله می‌کند؟ بعد از نیمه چه می‌شود؟ چرا بدون ZIP بازی شروع نمی‌شود؟

جلسه‌ی ۲: اولین برنامه: ربات را خط‌به‌خط تکان بده کدنویسی · LESSON = 1

هدف

با فرمان‌های ساده یک برنامه‌ی ترتیبی بنویسند و ببینند هر خط دقیقاً چه کاری می‌کند.

مفاهیم

- `forward(t)` و `backward(t)`: به مدت t ثانیه مستقیم جلو یا عقب. ربات با قطب‌نما جهتش را نگه می‌دارد.
- `turn_left(d)` و `turn_right(d)`: به اندازه‌ی d درجه در جا بچرخ.
- `wait(t)`: t ثانیه بایست. اول برنامه `wait(0.5)` می‌گذاریم تا حسگرها اولین عددشان را بگیرند.
- هر خط تا تمام نشود خط بعدی اجرا نمی‌شود. برنامه مثل یک دستور پخت است.

- در درس‌های ۱، ۲ و ۴ فقط ربات MOVER (پیش‌فرض ۳) حرکت می‌کند تا ربات‌ها به هم نخورند.

فعالیت

- در `chovgan_team/robot.py` بنویسید `LESSON = 1` ، پوشه‌ی chovgan_team و `team_name.txt` را ZIP کنید.
- Start → Import ZIP → Movement → Training. خروجی print را در Controller status → Esc ببینید.
- عددها را عوض کنید: `turn_left(45)` ، `forward(3)` . بعد از هر تغییر Reload code بزنید.
- چالش: ربات یک مستطیل بکشد و به جای اولش برگردد.

کد اصلی جلسه

```
self.wait(0.5)
self.turn_to(180)
self.forward(2)
self.turn_left(90)
self.forward(1.5)
self.turn_right(90)
self.backward(1)
```

ارزیابی

اگر `forward(2)` را به `forward(4)` تغییر دهیم چه می‌شود؟ چرا اول `wait(0.5)` لازم است؟

جلسه‌ی ۳: زیر فرمان‌ها: دو موتور و حلقه‌ی اصلی کدنویسی · LESSON = 2

هدف

بفهمند forward و turn در واقع دو عدد سرعت برای دو موتور هستند و حلقه‌ی step چطور زمان را جلو می‌برد.

مفاهیم

- دو موتور: left wheel motor و right wheel motor. سرعت هر کدام بین 10- و 10+. سرعت ۱۰ حدود ۱۴ متر بر ثانیه.
- هر دو مثبت و برابر = جلو. هر دو منفی = عقب. چپ منفی و راست مثبت = چرخش به چپ. دو مثبت نابرابر = قوس.
- `robot.step(32)` زمان بازی را ۳۲ میلی‌ثانیه جلو می‌برد. در کد پایه `self.tick()` همین کار را می‌کند و حسگرها را هم تازه نگه می‌دارد.
- جدول PLAN: هر ردیف تا یک زمان مشخص اجرا می‌شود (if t < until).

فعالیت

- `LESSON = 2` و اجرا. هر تغییر حرکت در Controller status چاپ می‌شود.
- در Robot Lab بخش Turning Test را امتحان کنید و رابطه‌ی سرعت چپ و راست با جهت چرخش را ببینید.
- یک ردیف به PLAN اضافه کنید که ربات یک دایره‌ی کامل بزند.

کد اصلی جلسه

```
self.left.setVelocity(5)    # موتور چپ
self.right.setVelocity(5)   # موتور راست: هر دو برابر = جلو
```

```
self.tick()                # ۳۲ میلی‌ثانیه جلو برو
```

ارزیابی

سرعت چپ ۶ و راست ۳: ربات به کدام سمت قوس می‌زند؟ چرا؟

جلسه‌ی ۴: همه‌ی حسگرها کدنویسی · LESSON = 3

هدف

هر حسگر را بخوانند، واحدش را بدانند و بگویند برای چه کاری مفید است.

مفاهیم

- `position()` :GPS جای ربات به متر. قطب‌نما: `heading_deg()` جهت ربات؛ ۰ = رو به +y، ۹۰ = رو به +x.
- چهار سونار: `distance("front")` و right، left، back. فاصله تا نزدیک‌ترین ربات، توپ یا مانع، تا ۱۰ متر.
- گیرنده‌ی توپ: جهت توپ نسبت به ربات [جلو، چپ، بالا] و قدرت سیگنال = ۱ / فاصله². پس فاصله = 1/√قدرت.
- `ball_angle()` : زاویه‌ی توپ نسبت به دماغه؛ مثبت = سمت چپ. `ball_position()` : جای توپ در زمین.
- attack_x: دروازه‌ای که باید به آن حمله کنیم. شبیه‌ساز می‌فرستد و نیمه‌ی دوم عوض می‌شود.

فعالیت

- `LESSON = 3` و اجرا. یک جدول بکشید: هر حسگر، عددی که دیدید، واحد، کاربرد.
- در Robot Lab → Open live sensor practice همان حسگرها را زنده ببینید و با خروجی خودتان مقایسه کنید.
- توپ را در ذهن جابه‌جا کنید: اگر `ball_angle()` منفی ۹۰ باشد توپ کجاست؟

کد اصلی جلسه

```
x, y = self.position()
h = self.heading_deg()
front = self.distance("front")
d = self.ball_distance()
a = self.ball_angle()
```

ارزیابی

قدرت سیگنال توپ ۰.۰۴ است. توپ چند متر دور است؟ (جواب: ۵ متر)

جلسه‌ی ۵: قطب‌نما و چرخش دقیق کدنویسی · LESSON = 4

هدف

با خطای زاویه و کنترل تناسبی ربات را دقیقاً رو به یک جهت بچرخانند.

مفاهیم

- خطای زاویه = هدف − جهت فعلی، همیشه بین 180- و 180+ (کوتاه‌ترین راه).
- کنترل تناسبی (P): هرچه خطا بیشتر، تندتر بچرخ. نزدیک هدف آرام.

- ترمز (D): اگر خیلی تند می‌چرخیم، کمی کم کن تا از هدف رد نشویم.
- `turn_to(۰)` رو به +x، `turn_to(۹۰)` رو به +y، `turn_to(۱۸۰)` رو به -x، `turn_to(-90)` رو به -y. در آزمایش ما خطا کمتر از ۳ درجه بود.
- چون هر ضلع مربع جهت مطلق دارد، خطای چرخش‌ها روی هم جمع نمی‌شود.

فعالیت

- `LESSON = 4` و اجرا. heading بعد از هر turn_to را با هدف مقایسه کنید.
- در `chovgan_robot.py` ضریب ۴.۰ تابع turn_to را ۱.۰ و بعد ۱۰ کنید و فرقش را ببینید.
- چالش: یک شش‌ضلعی با turn_to بکشید.

کد اصلی جلسه

```
for heading in (180, -90, 0, 90):
    self.turn_to(heading)
    self.forward(1.5)
```

ارزیابی

جهت فعلی ۱۷۰ و هدف 170- است. خطای زاویه چند درجه و ربات به کدام سمت بچرخد؟

جلسه‌ی ۶: GPS و رفتن به یک نقطه کدنویسی · LESSON = 5

هدف

ربات را با GPS و قطب‌نما به هر نقطه‌ی زمین بفرستند و گشت بزنند.

مفاهیم

- جهت نقطه‌ی هدف (tx, ty) از جای فعلی (x, y): `atan2(ty - y, tx - x)` .
- drive_to: اول رو به نقطه بچرخ، بعد جلو برو، نزدیک نقطه آرام شو و در ۱.۵ متری بایست.
- نیمه‌ی خودی سمت مخالف attack_x است: attack_sign-. پس همین کد بعد از عوض شدن نیمه هم درست است.
- در آزمایش ما هر چهار ربات به همه‌ی نقطه‌ها رسیدند و بعد از نیمه سمت درست را گشت زدند.

فعالیت

- `LESSON = 5` و اجرا. پیام arrived هر ربات را در Controller status ببینید.
- یک نقطه‌ی سوم به گشت اضافه کنید.
- چالش: هر چهار ربات یک خط افقی روی خط وسط بسازند.

کد اصلی جلسه

```
home = -self.attack_sign          # نیمه‌ی خودی
self.drive_to(home * 13, y)
self.drive_to(home * 31, y)
```

ارزیابی

ربات در (0, 0) است و رو به x+. نقطه‌ی (10, 0) در چه جهتی است و ربات اول چند درجه باید بچرخد؟

جلسه‌ی ۷: سونار و جلوگیری از برخورد کدنویسی · LESSON = 6

هدف

با if و else روی عدد سونار تصمیم بگیرند: جلو برو یا بایست و بچرخ.

مفاهیم

- `distance("front")` کمتر از SAFE متر = چیزی جلوی ماست.
- مقایسه‌ی چپ و راست: به سمتی بچرخ که سونارش جای بیشتری نشان می‌دهد.
- ربات‌های دیگر و توپ هر دو مانع حساب می‌شوند.

فعالیت

- `LESSON = 6` و اجرا. پیام obstacle هر ربات را بخوانید.
- SAFE را ۱.۵ و ۵ کنید. کدام بهتر است؟
- چالش: اگر پشت ربات هم بسته بود (`distance("back")`) چه کند؟

کد اصلی جلسه

```
if self.distance("front") > 3.0:
    self.set_motors(5, 5)
    self.tick()
elif self.distance("left") >= self.distance("right"):
    self.turn_left(60)
else:
    self.turn_right(60)
```

ارزیابی

سونار جلو ۲، چپ ۸ و راست ۳ است. ربات چه می‌کند؟

جلسه‌ی ۸: توپ: پشت توپ برو و به دروازه هل بده کدنویسی · LESSON = 7

هدف

جای توپ را حساب کنند و توپ را به سمت دروازه‌ی حریف هل بدهند، نه به هر سمتی.

مفاهیم

- ربات فرمان «شوت» ندارد. توپ را با راندن ربات به سمت آن هل می‌دهیم.
- اگر مستقیم به توپ بزنیم، توپ هر طرف که رو کرده‌ایم می‌رود؛ حتی به دروازه‌ی خودی.
- جهت توپ (bx, by) به دروازه: `atan2(0 - by, attack_x - bx)` . نقطه‌ی پشت توپ ۴ متر در خلاف همین جهت است.
- وقتی پشت توپ رسیدیم: turn_to رو به دروازه و forward با سرعت ۱۰.

فعالیت

- `LESSON = 7` و اجرا (فقط ربات ۳ بازی می‌کند).
- فاصله‌ی ۴ متر را ۲ و ۶ کنید و نتیجه را مقایسه کنید.
- در Training → Shooting همین کد را امتحان کنید.

کد اصلی جلسه

```
bx, by = self.ball_position()
to_goal = math.atan2(0 - by, self.attack_x - bx)
behind = (bx - 4 * math.cos(to_goal), by - 4 * math.sin(to_goal))
self.drive_to(*behind)
self.turn_to(math.degrees(to_goal))
self.forward(1.5, speed=10)
```

ارزیابی

توپ در (10, 0) است و attack_x = -52.5. نقطه‌ی پشت توپ کجاست؟ (جواب: (14, 0))

جلسه‌ی ۹: پیام تیمی: چه کسی سراغ توپ برود کدنویسی · LESSON = 8

هدف

با فرستادن و خواندن پیام JSON کار را بین ربات‌ها تقسیم کنند.

مفاهیم

- team emitter پیام متنی را فقط به سه هم‌تیمی می‌رساند. حریف نمی‌شنود. حداکثر ۴۰۹۶ بایت.
- `json.dumps` یک دیکشنری را به متن تبدیل می‌کند و `json.loads` برعکس. `self.inbox` پیام‌های رسیده است.
- هر ربات فاصله‌اش تا توپ را می‌فرستد. فقط نزدیک‌ترین سراغ توپ می‌رود. دروازه‌بان نامزد نیست.

فعالیت

- `LESSON = 8` و اجرا. ببینید فقط یک ربات سراغ توپ می‌رود.
- به پیام نقش ربات را هم اضافه کنید.
- چالش: دومین ربات نزدیک به‌جای ایستادن، ۸ متر پشت توپ منتظر بماند.

کد اصلی جلسه

```
self.team_emitter.send(json.dumps({"robot": self.name, "dist": mine}))
for msg in self.inbox:
    distances[msg["robot"]] = msg["dist"]
```

ارزیابی

اگر همه‌ی ربات‌ها پیام بفرستند ولی هیچ‌کدام inbox را نخوانند چه می‌شود؟

جلسه‌ی ۱۰: تیم کامل: چهار نقش کدنویسی · LESSON = 0

هدف

کد تیم کامل را بخوانند و برای هر نقش اولین بهبود را بدهند.

مفاهیم

- `robot1.py` دروازه‌بان: جلوی دروازه‌ی خودی می‌ماند و با توپ چپ و راست می‌رود؛ اگر توپ نزدیک شد دورش می‌کند.
- `robot2.py` مدافع: وسط نیمه‌ی خودی؛ اگر توپ نزدیک‌تر از ۹ متر شد سراغش می‌رود.
- `robot3.py` مهاجم: پشت توپ و هل به دروازه. `robot4.py` مهاجم کناری: وقتی ۳ نزدیک‌تر است، جلوتر و آن طرف زمین منتظر می‌ماند.
- `LESSON = 0` یعنی تیم کامل. در آزمایش ما مقابل تیم ایستای تمرینی ۱ بر ۰ برد و مقابل تیم تمرینی رسمی ۰ بر ۱ باخت.

فعالیت

- `LESSON = 0` و در Training مقابل حریف Baseline بازی کنید.
- GOAL_OFFSET دروازه‌بان و STRENGTH_ENGAGE مدافع را عوض کنید و اثرش را ببینید.
- یک ویژگی از درس‌ها (مثلاً سونار درس ۶) را به یکی از نقش‌ها اضافه کنید.

ارزیابی

نقش هر ربات را با یک جمله توضیح دهید. کدام نقش بیشترین اثر را روی گل خوردن دارد؟

جلسه‌ی ۱۱: راهبرد، آزمایش و تابلوی مربی (U14) پروژه · LESSON = 0

هدف

با آزمایش‌های تکرارپذیر کد تیم را بهتر کنند. U14 تابلوی مربی را به کد وصل کند.

مفاهیم

- یک تغییر در هر بار، بعد آزمایش. نتیجه را یادداشت کنید (گل زده، گل خورده، مشکل).
- Training پنج حالت دارد. Retry همان کد را دوباره اجرا می‌کند و Reload code کد ویرایش‌شده را می‌خواند.
- U14: پیام 1 `chovgan.coach.plan.v` در `self.coach_playbook` (BALANCED، ATTACK یا DEFEND). در U19 این پیام نمی‌آید.

فعالیت

- سه مسابقه‌ی تمرینی مقابل Baseline؛ جدول نتیجه بکشید.
- U14: اگر coach_playbook برابر DEFEND بود، مهاجم‌ها در نیمه‌ی خودی بمانند.
- یک اشکال واقعی را با print در Controller status پیدا و رفع کنید.

کد اصلی جلسه

```
if self.coach_playbook == "DEFEND":
    home = -self.attack_sign * 20
    self.drive_to(home, 0)
```

ارزیابی

گزارش کوتاه: چه چیزی را عوض کردید، چرا، و نتیجه چه شد؟

جلسه‌ی ۱۲: تحویل و مسابقه‌ی آزمایشی مسابقه · LESSON = 0

هدف

ZIP تحویلی درست بسازند و در یک مسابقه‌ی آزمایشی شرکت کنند.

مفاهیم

- `LESSON = 0` باشد. ZIP فقط `chovgan_team/` و `team_name.txt` را در سطح اول دارد.
- فایل‌های مجاز در ZIP: .py، .json، .md، .txt و .png. ماژول‌های os، sys و socket و کتابخانه‌های بیرونی مثل numpy بسته‌اند.
- خطاهای رایج STRATEGY: MISSING_ENTRYPOINT، IMPORT_ERROR، UNKNOWN_DEVICE، PYTHON_SYNTAX_ERROR.
- محل و مهلت تحویل در کانال کمیته فنی اعلام می‌شود.

فعالیت

- ZIP را بسازید و در Competition → STRATEGY مطمئن شوید VALID است.
- مسابقه‌ی آزمایشی کلاس: هر تیم مقابل دو تیم دیگر.
- بازبینی: یک چیز که کار کرد، یک چیز که کار نکرد، قدم بعدی.

ارزیابی

روبریک: اجرا و عملکرد ۴۰٪، درک مفاهیم ۳۰٪، کار تیمی و پیام‌ها ۱۵٪، تحلیل و بهبود ۱۵٪.

نام‌هایی که باید دقیق بنویسید

```
"gps"                  "compass"
"ball receiver"        "supervisor receiver"
"team emitter"         "team receiver"
"distancesensor front" "distancesensor back"
"distancesensor left"  "distancesensor right"
"left wheel motor"     "right wheel motor"

TIME_STEP = 32 ms   |  motor speed: -10 .. +10   |  team message <= 4096 bytes
```