University of Idaho

STANDARDBOT CRASH COURSE

DAWSON BURGERS
CDA ROBOTICS 1

RESOURCES

THESE WILL BE YOUR FRIENDS

- 1. Standard Bots Documentation
  - https://docs.standardbots.com/docs/latest
- 2. Dawson (unfortunately, no offense)
- 3. If you choose to pin your standardbots python package version - version = “z"
  - https://pypi.org/project/standardbots/2.20250128.54/#history
- Presentation by Sarah Davis (I will give to you)
- A way to control python env – I recommend UV (or however you are comfortable)
  - Very good docs, 1 command to set up: https://docs.astral.sh/uv/

A NOTE ON PINNING THE VERSION

SBOTS ARE GREAT OUTSIDE OF ONE TINY SUPER SMALL INCONVENIENT DETAIL

- Sbot updates their python package and code a lot! Buuuuuuuttt not always their docs in synchrony…
- This leaves you with a few options:
  - 1. Install normally and read through a combination of the sbot docs and python package yourself to see definitions/methods/etc and work to get them working. (**Package name:** standardbots)
    - I will not be as helpful here, not up to date on how they're doing things but I'll try
    - If you do this option and have an LSP/intellisense esk plugin, use it. It will make this a lot less awful.
  - 2. Install the pinned version I know that works, and it will work with the samples in this powerpoint.
    - Risk here is that you will need to parse through the package, docs won't help you here as much

SBOT CODE WALKTHROUGH

FROM THEIR DOCS

- No matter what you do, you will use the sdk to open a connection with the sbot
- To get the 2 variables you will need to connect:
  - 1. Find the tablet associated with you sbot
  - 2. Go to the settings in the bottom left hand side
  - 3. Open the developer api config
  - 4. copy down the long string, you will need that to connect
    - Will be long like “pcabdkvh-ktr69i-bzh47wpj-alcfy” or something similar
  - 5. note the robots ip, you will connect with http://10.8.4.x:3000
    - DO NOT forget the port 3000, make sure you have the correct token

SBOT CODE WALKTHROUGH

FROM THEIR DOCS

- Once you have connection variables, we can connect
- Everything you will do will happen using the sdk connection as seen in the second part of the code

python main.py

```python
from standardbots import StandardBotsRobot

sdk = StandardBotsRobot(
  url='https://mybot.sb.app',
  token='token',
  robot_kind=StandardBotsRobot.RobotKind.Live,
)

# Optional, manage creation and deletion of sdk
with sdk.connection():
  [... your logic here ...]
```

Live Robot Copy

SBOT CODE WALKTHROUGH

FROM THEIR DOCS

- This is what it starts to look like when you do more complicated things, for instance: gripper control
- It operates in this weird madness of nested variables that makes sense once you get the hang of it (reading through the source code helps here, but is a lot)
- Make sure to do with an sdk.connection(). Then you need to have a response object with the appropriate params.
- I believe the gripper is 2Fg14, don’t quote me on that
- Take a look at their docs, but there is a chance that they will not be accurate

python cURL POST /api/v1/equipment/end-effector/control

```python
from standardbots import models, StandardBotsRobot

sdk = StandardBotsRobot(
    url='https://mybot.sb.app',
    token='token',
    robot_kind=StandardBotsRobot.RobotKind.Live,
)

position = 10.0

with sdk.connection():
    response = sdk.equipment.control_gripper(
        models.GripperCommandRequest(
            kind=models.GripperKindEnum.Onrobot2Fg14,
            onrobot_2fg14=models.OnRobot2FG14GripperCommandRequest(
                control_kind=models.OnRobot2FG14ControlKindEnum.Move,
                target_grip_width=models.LinearUnit(
                    unit_kind=models.LinearUnitKind(
                        models.LinearUnitKind.Millimeters
                    ),
                    value=position,
                ),
                grip_direction=models.LinearGripDirectionEnum(
                    models.LinearGripDirectionEnum.Internal
                ),
            ),
        )
    )

try:
    data = response.ok()
except Exception:
    print(response.data.message)

```

Live Robot Copy

SBOT CODE WALKTHROUGH
FROM THEIR DOCS

```python
from standardbots import models, StandardBotsRobot

sdk = StandardBotsRobot(
  url='https://mybot.sb.app',
  token='token',
  robot_kind=StandardBotsRobot.RobotKind.Live,
)

with sdk.connection():
  sdk.movement.brakes.unbrake().ok()
  sdk.movement.position.move(
    position=models.Position(
      unit_kind=models.LinearUnitKind.Meters,
      x=0.1,
      y=0.2,
      z=1.0,
    ),
    orientation=models.Orientation(
      kind=models.OrientationKindEnum.Quaternion,
      quaternion=models.Quaternion(0.0, 0.0, 0.0, 1.0),
    ),
  ).ok()
```

main.py

Live Robot Copy

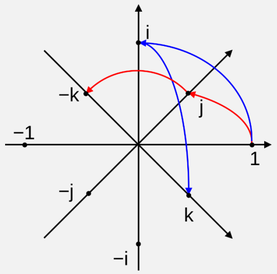
- This is what it starts to look like for movements
- Not sure how well this works, I have not had time to test it quite yet
- A lot of this is going you reading through code+the docs to see what is accurate and how to properly format things
  - I HIGHLY recommend some sort of intellisense or LSP to help with this. It shows you function definition from the packages if you use it with you python env.
- Anyways you have the basics, I will provide some example code for the **PINNED VERSION** earlier in the slides

A QUICK INTERMISSION

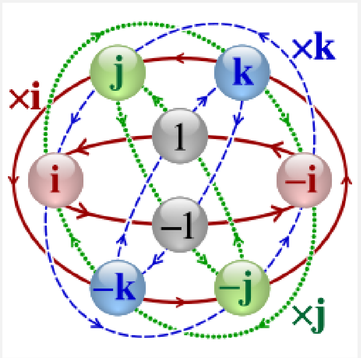
DIFFERENCES

- Both 6-Axis Cobots
- **FANUC:**
  - Had to build custom python
  - Bigger
  - A bit more stable, sbot is kinda just rolling on wheels
  - More sensitive force sensors
- **SBOT:**
  - Built in camera in the EOAT
  - Python support out of the box with a package
  - Way better teaching pendant
  - Weird quarks, relatively new company
  - Slightly smaller
  - Be careful when programming/around the SBOT in general. It does take a fair bit more force for it to be stopped when it hits an object
  - Quaternions.....

A WHAT NOW?



ij = k
ji = −k
ij = −ji



- In modern terms, quaternions form a four-dimensional associative normed division algebra over the real numbers, and therefore a ring, also a division ring and a domain. Now instead of Yaw, pitch, and roll you will have x, y, z, and w.
  - https://en.wikipedia.org/wiki/Quaternion
- So, there is going to be more complications/a learning curve with the quaternions. They are easy to translate one way, not the other. It sucks to get used to unless you are a math whiz, but it is objectively better than how FANUC does it. The turn 1,2,3 system is more ambiguous than unambiguous quaternion coordinates.
- For this I recommend –
  - Moving the gripper to the desired position, using the read_position(or something similar that allows you to see the current pos and quat), and write that down. Just use that.
  - Add more positions if need, set conditions to check if in a certain range and switch/match, etc
- Of if you're like really good at math, cool, have fun doing that because I do not know it well enough yet.

SOME MORE WEIRDNESS

- You need to make sure that you are using the proper units
  - Sbot sdk is pretty stickler about this
  - Ex: accidentally being in mm when set to meters
  - It is a lot different than programming the FANUC
  - Units can be: mm, m, radian, degree, etc. Look though the docs/code you can set what you want. You will see examples in these slides
  - Make sure you double check everything, some errors can happen at this level
  - If you choose to use a newer version, PIN YOU PACKAGE. I had mine update on me once and things tend to break. You will want to do this especially if you have to figure out some of the quarks and don’t want to re-do that. Ask me….

SBOT CODE WALKTHROUGH
FROM MY CODE

- Pinned version: 2.20241120.1
- Connection, not much has changed
- Looks the same
- Make sure you have the right ip address, and the correct token

```python
import cv2 as cv
import numpy as np
from standardbots import StandardBotsRobot, models


# dave
sdk = StandardBotsRobot(
    url="http://172.29.208.19:3000",
    token="pcabdkvh-ktr69i-bzh47wpj-alcfy",
    robot_kind=StandardBotsRobot.RobotKind.Live,
)
```

SBOT CODE WALKTHROUGH
FROM MY CODE

- Pinned version: 2.20241120.1
- **This will be your bread and butter** until you get comfortable
- This method gets the position of the robot
  - Joints, cartesian position, orientation (in quaternions)
- Can just load this up in a simple program, hand guide the robot, record positions, etc.
  - To hand guide the robot, touch the silver button on the left of the EOAT near the OnGripper quick connect

```python
def get_position_info():
    with sdk.connection():
        sdk.movement.brakes.unbrake().ok()
        response = sdk.movement.position.get_arm_position()

        try:
            data = response.ok()
            j_1, j_2, j_3, j_4, j_5, j_6 = data.joint_rotations
            position = data.tooltip_position.position
            orientation = data.tooltip_position.orientation
            joints = data.joint_rotations

            print(f"Joints: {joints}")
            print(f"Got Position: {position}")
            print(f"Got orientation: {orientation}")

            return j_1, j_2, j_3, j_4, j_5, j_6

        except Exception:
            print(response.data.message)
```

SBOT CODE WALKTHROUGH
FROM MY CODE

- Pinned version: 2.20241120.1
- Cartesian move method
- A Sbot quark: when you move, you first need to unbrake. The robot lives in a default state of braked when not actively moving. *Will not move unless you unbrake.*
- You can see in the code I have it set to meters. I measure in mm or m, convert. When this method is called, plug in the 3 variables in m units
- Orientation was 1 of 2 I had, other was (0.0, 0.0, 0.0, 0.1). Play with this to get a feel, there is math you can do, fall back to hand guiding.

```
def move_robot_cartesian(x, y, z):
    with sdk.connection():
        sdk.movement.brakes.unbrake().ok()
        sdk.movement.position.move(
            position=models.Position(
                unit_kind=models.LinearUnitKind.Meters,
                x=x,
                y=y,
                z=z,
            ),
            orientation=models.Orientation(
                kind=models.OrientationKindEnum.Quaternion,
                quaternion=models.Quaternion(
                    -0.50344496073135,
                    -0.48649383205875135,
                    0.5136321065685251,
                    -0.4960332468544624,
                ),
            ),
        ).ok()
```

SBOT CODE WALKTHROUGH
FROM MY CODE

```python
def move_robot_joint(j1, j2, j3, j4, j5, j6):
    with sdk.connection():
        sdk.movement.brakes.unbrake().ok()
        arm_rotations = models.ArmJointRotations(joints=(j1, j2, j3, j4, j5, j6))
        # Log to ensure the values are correct
        position_request = models.ArmPositionUpdateRequest(
            kind=models.ArmPositionUpdateRequestKindEnum.JointRotation,
            joint_rotation=arm_rotations,
        )
        sdk.movement.position.set_arm_position(position_request).ok()
```

- Pinned version: 2.20241120.1
- Joint move method
- Performs just like you know and love, same “quirks” (things you are not used to) with joint moves ie limit errors and being cranked too far on certain combinations (j4+j5 etc).

SBOT CODE WALKTHROUGH
FROM MY CODE

- Pinned version: 2.20241120.1
- Camera example method
- Just a general look at what the programming looks like. Will be a mix of general python stuff and some sbot stuff
- Uses the built in camera
- A bit more complicated, has some extra steps for processing with opencv

```python
def capture_image():
    body = models.CameraFrameRequest(
        camera_settings=models.CameraSettings(
            brightness=0,
            contrast=50,
            exposure=250,
            sharpness=50,
            hue=0,
            whiteBalance=4600,
            autoWhiteBalance=True,
        )
    )

    with sdk.connection():
        response = sdk.camera.data.get_color_frame(body)

        print(f"Result Data: {response.data}")
        response.ok()
        raw_data = response.response.data

        # extract
        base64_data = raw_data.decode().split(",")[1]

        # decode
        image_data = base64.b64decode(base64_data)

    # convert to numpy array
    np_data = np.frombuffer(image_data, np.uint8)

    # read image
    img = cv.imdecode(np_data, cv.IMREAD_COLOR)

    return img
```

SBOT CODE WALKTHROUGH
FROM MY CODE

- Pinned version: 2.20241120.1
- Gripper method
- This is how I chose to tackle the gripper
- By far the most complicated thing for Sbots outside of quaternions
- Pay close attention to whether you are doing and inward or outward direction.
- You need: force in N, target size in mm or m
- Then I just have a wrapper to make it easier to call

```python
def gripper_request(WIDTH, FORCE):
    with sdk.connection():
        response = sdk.equipment.control_gripper(
            models.GripperCommandRequest(
                kind=models.GripperKindEnum.Onrobot2Fg14,
                onrobot_2fg14=models.OnRobot2FG14GripperCommandRequest(
                    grip_direction=models.LinearGripDirectionEnum.Inward,
                    target_grip_width=models.LinearUnit(
                        value=WIDTH, unit_kind=models.LinearUnitKind.Meters
                    ),
                    target_force=models.ForceUnit(
                        value=FORCE,
                        unit_kind=models.ForceUnitKind.Newtons,
                    ),
                    control_kind=models.OnRobot2FG14ControlKindEnum.Move,
                ),
            )
        )

    try:
        data = response.ok()

    except Exception:
        print(response.data.message)


def gripper_command(STRING):
    if STRING == "OPEN":
        print("OPENING GRIPPER")
        gripper_request(0.11, 10.0)
    if STRING == "CLOSE":
        print("CLOSING GRIPPER")
        gripper_request(0.078, 10.0)
```

SBOT CODE WALKTHROUGH
FROM MY CODE

- Pinned version: 2.20241120.1
- Last bit of code snippets because why not
  - Home position, joint move example, random snippets from when I took the class
- This is a lot better than where I started
- To set up the LSP/intellisense just google/youtube it, very easy and helpful.

```python
def place_dice(dice_num, dice_coords, dice_stack_num):
    DICE_HEIGTH = 0.080
    STACK_HEIGHT = 0.0376
    ABOVE_DICE_PLACE_HEIGHT = -0.3789

    x_coord_target = dice_coords[dice_num][0]
    y_coord_target = dice_coords[dice_num][1]
    z_coord_target = dice_coords[dice_num][2]

    # dice stack location 1, z adjusts
    x_dice_stack_1 = 0.6483
    y_dice_stack_1 = 0.5055
    z_dice_stack_1 = STACK_HEIGHT - (DICE_HEIGTH * (dice_num + 1))

    # dice stack location 2, z adjusts
    x_dice_stack_2 = 0.7963
    y_dice_stack_2 = 0.4966
    z_dice_stack_2 = STACK_HEIGHT - (DICE_HEIGTH * (dice_num - 5))

    if dice_stack_num == 1:
        # move to stack
        gripper_command("OPEN")
        move_robot_cartesian(x_dice_stack_1, y_dice_stack_1, z_dice_stack_1 + 0.08)
        move_robot_cartesian(x_dice_stack_1, y_dice_stack_1, z_dice_stack_1)

        # close gripper and move up
        gripper_command("CLOSE")
        move_robot_cartesian(x_dice_stack_1, y_dice_stack_1, z_dice_stack_1 + 0.08)

        # ABOVE WORK AREA
        move_robot_joint(
            -1.2692447900772095,
            -0.41903451881371573,
            -1.6317762802063975,
            0.5168988555844456,
            1.5609856843948395,
            -2.834940195083619,
        )

        # pre dice place move
        move_robot_cartesian(x_coord_target, y_coord_target, ABOVE_DICE_PLACE_HEIGHT)

        # move down to board
        move_robot_cartesian(x_coord_target, y_coord_target, z_coord_target)
        gripper_command("OPEN")
        move_robot_cartesian(x_coord_target, y_coord_target, ABOVE_DICE_PLACE_HEIGHT)

    if dice_stack_num == 2:
        # move to stack
        gripper_command("OPEN")
        move_robot_cartesian(x_dice_stack_1, y_dice_stack_1, z_dice_stack_2 + 0.2)
        move_robot_cartesian(x_dice_stack_2, y_dice_stack_2, z_dice_stack_2 + 0.08)
        move_robot_cartesian(x_dice_stack_2, y_dice_stack_2, z_dice_stack_2)
```

```python
print("HOMING")
#get_position_info()
move_robot_joint(0.0, 0.0, -1.5, 0.0, 1.5, -3.14)
#get_position_info()
```

- Snippet from a program that takes dice from a stack of 2 towers, places them into a grid, then takes an image for camera-robot coordinates calibration.
- Just an example of how you can do things if you follow something similar to my code examples
- Pro tip: figure out all of the standardbot stuff in methods, so you can clean up your main loop.

Q/A

THANK YOU FOR YOUR ATTENTION, SORRY YOUR INTRODUCTION IS LIKE THIS.... GOODLUCK