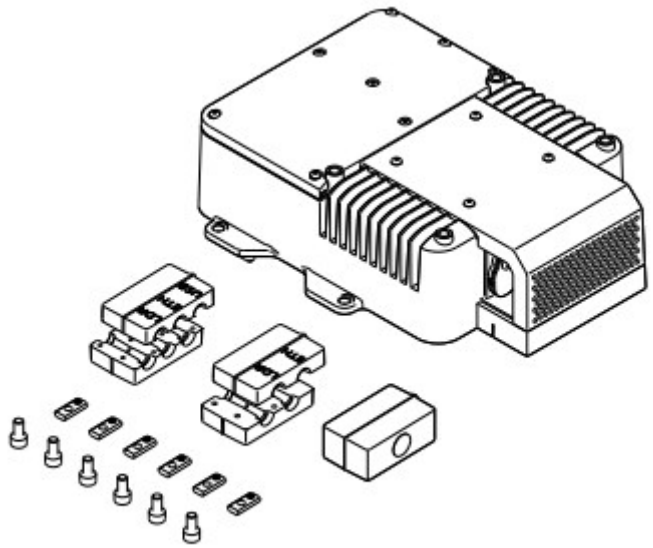
Basics

Basics

CORE I/O



Network

Connecting through SPOT

The Spot CORE I/O can also be set up to use Spot as a router in order to access the internet if Spot is connected. When the default route of the Spot CORE I/O is set to 192.168.50.3 (Spot), it can access the same networks as the robot. See the knowledge article Spot CORE: Accessing the Internet through Spot for more details (this article is for the legacy Spot CORE, but will also apply to the Spot CORE I/O).

The Spot CORE I/O does not have Cockpit. Network configuration sections referring to Cockpit should be done through the terminal instead of using the "nmcli" command line tool. The following commands will set the Spot CORE I/O's Gateway to Spot:

```
sudo nmcli connection modify eth-robot ipv4.gateway "192.168.50.3"
sudo nmcli c up eth-robot
```

Bare Minimum code to control spot

DISCLAIMER:

This documentation is still a work in progress. I will find ways to make this article and the code provided more efficient. If there are any errors you encounter please reach me by my discord: bookis.

Introduction:

Once you have established a working connection with SPOT and run a few of the Boston Dynamic (BD) scripts, you will probably want to write your scripts for Tape Measure. This article will show you how to create a simple standing script which will hopefully help you understand SPOT programming.

When writing your scripts for SPOT, you need to ensure your code has three things:

1. The IP, username, and password for SPOT.
2. Establish the E-Stop.
3. Claim the lease.

Imports:

```
from bosdyn.geometry import EulerZXY
import bosdyn.client
from bosdyn.client.robot_command import RobotCommandClient, blocking_stand #Ignore this for
now !!
from bosdyn.client.robot_command import RobotCommandBuilder
```

The bosdyn.client library is what will carry the bulk of the commands needed for you to complete the three aforementioned requirements to control SPOT.

Initializing the robot:

```
sdk=bosdyn.client.create_standard_sdk('test')
robot= sdk.create_robot('ip address')
robot.authenticate("username","password") #ASK DWIGHT FOR THE LOG IN INFORMATION AND IP
```

```
ADDRESS
state_client=robot.ensure_client('robot-state')
```

The code above is the basics of initializing the robot. The first line is where you access the sdk needed to reach the create_robot object you need not worry too much of what you put into the parameter (at least from what I understand). After accessing the create_standard_sdk object you need to call the "create_robot" object within the sdk and assign it to a variable (In my case its called "robot"). This variable is where you will be conducting a bulk of the commands needed to access SPOT's data and eventually be able to control him. You can test that you have access to him by typing the command below.

```
print(state_client.get_robot_state())
```

Now if you only need to get sensor data, this is the furthest extent of what you need to acquire such data. However, if you would like to physically control SPOT you need to initialize the estop and then acquire the lease for the SPOT.

Estop:

BD placed safety protocols that need to be done before you are allowed to physically control SPOT. The first step is initializing the estop.

```
estop_client=robot.ensure_client('estop')
estop_endpoint=
bosdyn.client.estop.EstopEndpoint(client=estop_client,name='my_estop',estop_timeout=9.0)
estop_endpoint.force_simple_setup()
estop_keep_alive=bosdyn.client.estop.EstopKeepAlive(estop_endpoint)
estop_client.get_status()
```

This is the code that's needed to initialize the estop. BD has it set up where you only need to call the get_status() object to be called to ensure that the estop has been initialized. To do that you need to ensure that there is an estop endpoint established and you need to have an estop client variable.

Acquiring the Lease:

The code below is what is needed to acquire the lease. Like with the estop, you only need to call the acquire() object inside of the lease client to get the lease.

NOTE: This will not forcibly take a lease from another device. If there is another device that is taking control of the robot make sure to release the lease in that device BEFORE you forcibly take the lease.

```
lease_client=robot.ensure_client('lease')
lease_client.list_leases()
lease=lease_client.acquire()
lease_keep_alive=bosdyn.client.lease.LeaseKeepAlive(lease_client)
lease_client.list_leases()
```

Making SPOT Stand:

Now after completing the previous steps, you should have everything you need to actually control SPOT. For the sake of simplicity, I condensed acquiring the lease and estop initializing into functions.

```
estopInilization() #This is what you did in the estop step
acquireLease() #This is what you did in the acquiring the lease step
robot.power_on(timeout_sec=20)
robot.is_powered_on()
robot.time_sync.wait_for_sync() # You need to sync the time thats on SPOT with the system time
on your device.
```

This next bit of code is going to be how you usually issue commands to SPOT. There are three steps that you need to do to issue commands to SPOT.

1) Create a command client.

2) Create the command.

3) Issue the command to the command client.

You will repeat steps 2 and 3 as necessary if you are trying to issue different commands.

```
command_client=robot.ensure_client(RobotCommandClient.default_service_name) #Step 1

cmd=RobotCommandBuilder.synchro_stand_command(footprint_R_body=footprint_R_body) #Step 2
command_client.robot_command(cmd) #Step 3
```

It's important to keep in mind that most of SPOTs commands will have a similar format. Whether it is to move or get a camera feed you need to create a client, request, then execute.

Full Code:

```
from bosdyn.geometry import EulerZXY
import bosdyn.client.robot_command
from bosdyn.client.robot_command import RobotCommandClient
from bosdyn.client.robot_command import RobotCommandBuilder

footprint_R_body = EulerZXY(yaw=0.1, roll=0.0, pitch=0.0)
def intilizeBot ():
    sdk=bosdyn.client.create_standard_sdk('test')
    robot= sdk.create_robot('ASK DWIGHT FOR IP')
    robot.authenticate("USER","PASSWORD") #ASK DWIGHT FOR PASSWORD AND USERNAME
    state_client=robot.ensure_client('robot-state')
    state=state_client.get_robot_state()
    return robot


def estopInilization():
    estop_client=robot.ensure_client('estop')
    estop_client.get_status()
    estop_endpoint=
bosdyn.client.estop.EstopEndpoint(client=estop_client,name='my_estop',estop_timeout=9.0)
    estop_endpoint.force_simple_setup()
```

```
    estop_keep_alive=bosdyn.client.estop.EstopKeepAlive(estop_endpoint)
    estop_client.get_status()


def acquireLease():
    lease_client=robot.ensure_client('lease')
    lease_client.list_leases()
    lease=lease_client.acquire()
    lease_keep_alive=bosdyn.client.lease.LeaseKeepAlive(lease_client)
    lease_client.list_leases()



robot=intilizeBot()
estopInilization()
acquireLease()
robot.power_on(timeout_sec=20)
robot.is_powered_on()
robot.time_sync.wait_for_sync()
command_client=robot.ensure_client(RobotCommandClient.default_service_name)
robot.start_time_sync()
cmd=RobotCommandBuilder.synchro_stand_command(footprint_R_body=footprint_R_body)
command_client.robot_command(cmd)
```