# User Study 02 - RL Audio Notebook

Please click the folliwng two links to read the explanatory statrement and answer the pre-study questionnaire.

**Explanatory Statement:** https://drive.google.com/file/d/1-8npbW1wg_ABzBnnGa1dgEgCaYjDED8o/view?usp=sharing

**Pre-study Questionnaire:** https://forms.gle/GAU8xzekWKkTMDLVA (Participant ID Required)

# Setup

## Imports & Args

**Before starting this Notebook...**

1. Enable JupyterLab Dark. Under "settings" --> theme --> **"JupyterLab Dark"**
2. Sub the line of code below with the path on your device to: **./../RL_audio/notebooks**

In [ ]:
```
%cd /home/liamroy/Documents/PHD/repos/RL_audio/notebooks

# %cd /Users/liamroy/Documents/Studies/Monash_31194990/PHD/repos/RL_audio/no

# %cd <add your path here and comment out the others>
```

In [ ]:
```
PWD = %pwd
```

In [ ]:
```
# You will need to install:
# --> pygame   (see this webpage ~ https://www.pygame.org/wiki/GettingStarte
# --> jupyterlab, numpy, termcolor, openpyxl, nbconvert-webpdf

# Either use:  sudo apt-get install <package_name>
#              python3 -m pip install <package_name>
#              conda install -c conda-forge <package_name>

# Example using conda:
# --> conda install -c conda-forge <package_name>
#                                  jupyterlab or notebook or
#                                  numpy
#                                  termcolor
#                                  openpyxl
#                                  nbconvert-webpdf
#
```

```
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# IMPORTS
import os
import shutil
import time
import numpy as np
import random
import argparse
import linecache

from scripts import audio_control
from scripts import ucb1_algorithm as ucb1
from scripts import misc_helpers as mischelp

import sys

from termcolor import colored, cprint
# Termcolor guide: https://pypi.org/project/termcolor/


#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#  ARGUMENTS & PARSER (Save this code for scripts working with CLI)

# argParser = argparse.ArgumentParser()

# # Enter any valid integer value
# argParser.add_argument("-b", "--budg", required=False, help="select the bu

# # Enter a valid parameter discritization integer (must match sound library
# argParser.add_argument("-d", "--disc", required=False, help="select discri

# # Enter true if you would like to see hidden print log, including Q-tables
# argParser.add_argument("-p", "--prnt", required=False, help="show hidden p

# # To load and save, simply enter in the base filename such as "lastsave" d
# argParser.add_argument("-s", "--save", required=False, help="filename to s
# argParser.add_argument("-l", "--load", required=False, help="load Q-table
```

# Initializations

In [ ]:

```
# Parameter discritization
param_disc = 3

state_descriptions = ["Stuck      \t- robot needs your help", "Successful \t
num_of_states = len(state_descriptions) - 1 # Adding a minus 1 since the las
state_range = np.arange(num_of_states)


# CREATE SOUND LIBRARY A
# For library A, setup the array using libA
library_A = "libA"

# Create an array of size (N x N x N) where N = number of discretized regior
# number of discretized regions for each param --> i.e. if equals 3 then (0,
```

```
# ** must align with the discretization for selected sound library
sound_obj_array_A = np.ndarray((param_disc, param_disc, param_disc),dtype=ob

for param_1_range in range(param_disc):
        for param_2_range in range(param_disc):
                for param_3_range in range(param_disc):
                        sound_obj_array_A[param_1_range, param_2_range, para


# CREATE SOUND LIBRARY B
# For library B, setup the array using libB
library_B = "libB"

# Create an array of size (N x N x N) where N = number of discretized regior
# number of discretized regions for each param --> i.e. if equals 3 then (0,
# ** must align with the discretization for selected sound library
sound_obj_array_B = np.ndarray((param_disc, param_disc, param_disc),dtype=ob

for param_1_range in range(param_disc):
        for param_2_range in range(param_disc):
                for param_3_range in range(param_disc):
                        sound_obj_array_B[param_1_range, param_2_range, para
```

# MAIN STUDY

Welcome to this study's **Jupyter notebook**. In this work, we are developing strategies for improving human-robot interaction with nonverbal sounds (***beeps & boops***).

This study is best completed with **headphones**. Ensure your volume is on.

While a robot is working on a task, it can have many different internal states...

If the robot gets stuck behind an obstacle, the robot's internal state is: **Stuck**

Similarly, if the robot was able to reach it's goal, the robot's internal state is: **Successful**

If the robot is actively working on the task but has neither gotten stuck nor completed the task, the robot's internal state is: **Progressing**

In this notebook, you will be asked to run through **3 sections**. In each of these sections, a virtual robot will play a sound. Once you listen to the sound, you will be asked to select which robot state you think the virtual robot is in. You will have the options: **Stuck**, **Successful**, **Progressing** and **Not Sure**

In addition to each answer, you will also self-score how confident you are in your response, on a scale from 1 to 10.

This process will repeat several times as a learning algorithm is processing in the background. **If you have any questions, ask your study moderator**. Have fun!

# SECTION 1A

Start by entering your user ID.

**Click on the first cell below & hit 'shift + enter'...**

In [ ]:
```
current_user_ID_str = mischelp.get_user_ID(parent_dir=PWD, num_of_states=num
```

**Our first robot is named Jackal.**

Let's listen to **Jackal** make a few sounds to express itself.

For each sound, you will asked to select which robot state you think the robot is in.

**Click on the cell below & hit 'shift + enter'...**

In [ ]:
```
mischelp.get_user_accuracy(sound_obj_array=sound_obj_array_A, lib_str=librar
                           states_array=np.ndarray(num_of_states, dtype=obje
```

 Jackal Robot

# SECTION 1B

**Our next robot is named the Spot.**

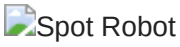
Let's listen to **Spot** make a few sounds to express itself.

You will notice **Spot** sounds slightly different to **Jackal**. For each sound, you will asked to select which robot state you think the robot is in.

**Click on the cell below & hit 'shift + enter'...**

In [ ]:
```
mischelp.get_user_accuracy(sound_obj_array=sound_obj_array_B, lib_str=librar
```

```
                                states_array=np.ndarray(num_of_states, dtype=obje
```




## Section 2

In section 2, we'll be listening to **Jackal** again.

Similar to before, **Jackal** make a few sounds to express itself, and you will asked to select which robot state you think the robot is in.

This process will repeat several times as a learning algorithm is processing in the background.

### Section 2X

**Click on the cell below & hit 'shift + enter'...**

In [ ]:
```
# SECTION 2X


time_step_2_str = ucb1.ucb1_algor(num_of_states=num_of_states, state_descrip
                                  current_user_ID_str=current_user_ID_str, sect
```

## Section 2O

**Click on the cell below & hit 'shift + enter'...**

In [ ]:
```
# SECTION 20


time_step_20_str = ucb1.ucb1_algor(num_of_states=num_of_states, state_descri
                                   current_user_ID_str=current_user_ID_str, sect
```


## Section 3A

We're nearly finished ~ **home stretch!**

Let's listen to **Jackal** express itself one last time.

For each sound, you will asked to select which robot state you think the robot is in.

**Click on the cell below & hit 'shift + enter'...**

In [ ]:
```
sect3_load_str = current_user_ID_str + "_sect20_step" + time_step_20_str
```

```
mischelp.get_user_accuracy(sound_obj_array=sound_obj_array_A, lib_str=librar
                           states_array=np.ndarray(num_of_states, dtype=obje
```


## Section 3B

Lastly, let's listen to **Spot** express itself one last time.

You will notice **Spot** sounds slightly different to **Jackal**. For each sound, you will asked to select which robot state you think the robot is in.

**Click on the cell below & hit 'shift + enter'...**

In [ ]:
```
mischelp.get_user_accuracy(sound_obj_array=sound_obj_array_B, lib_str=librar
                           states_array=np.ndarray(num_of_states, dtype=obje
```

## Save the Output

Run the following code block to save the output of this Jupyter Notebook.

**Click on the cell below & hit 'shift + enter'...**

In [ ]:
```
file_path_name = "user_data/user_" + current_user_ID_str + "/final_output"

cmd = "jupyter nbconvert --to webpdf --allow-chromium-download study_noteboc
if(os.system(cmd)):
        print("Error converting to .py")
        print(f"cmd: {cmd}")
```

## Closing Survey

Please click the folliwng link to answer a short post-study questionnaire.

**Pre-study Questionnaire:** https://forms.gle/K6RnncY82vSVdyE38 (Participant ID Required)

Thank you for completing this Jupyter Notebook.

### NOTES & DEBUG

**This section is not part of the survey.**

In [ ]:
```
# PILOTSET ARRAY VALUE SETTER

# State 0: Stuck - Pilot Set
manual_Qtable_state_0 = np.array([[[1., -1., -3.], [2., 0., -3.], [3., 2., -
                                                                  [[2., -1.,
                                                                  [[2., -1.,


print("State 0: Stuck")
print(manual_Qtable_state_0.shape, "\n")
print(manual_Qtable_state_0, "\n")



# State 1: Successful - Pilot Set
manual_Qtable_state_1 = np.array([[[-3., 0., 2.], [-3., 1., 3.], [-3., 0., 2
                                                                  [[-3., 0.,
                                                                  [[-3., 0.,


print("State 1: Successful")
print(manual_Qtable_state_1.shape, "\n")
```

```
print(manual_Qtable_state_1, "\n")



# State 2: Progressing - Pilot Set
manual_Qtable_state_2 = np.array([[[0., 3., 0.], [-3., 2., -3.], [-3., 1., -
                                                                  [[0., 5.,
                                                                  [[0., 4.,

print("State 2: Successful")
print(manual_Qtable_state_2.shape, "\n")
print(manual_Qtable_state_2, "\n")



np.save("arrays/pilotset_st0.npy", manual_Qtable_state_0)
np.save("arrays/pilotset_st1.npy", manual_Qtable_state_1)
np.save("arrays/pilotset_st2.npy", manual_Qtable_state_2)
```

Creating buttons and widgets: https://medium.com/@technologger/how-to-interact-with-jupyter-33a98686f24e

In [ ]:

In [ ]:
```
%whos
```