Project Plan

2.1 Project Management/Tracking Procedures

Our group plans to utilize an agile project management style. With the main goals of the project being to successfully demonstrate embedded machine learning (ML) on an interesting application and to make recommendations for incorporating embedded ML in a course for the CPR E department, we will be able to break down the implementation of a walking robot, our selected application, into tasks that can readily be completed in an agile environment. We will utilize Github to track our progress throughout the semester.

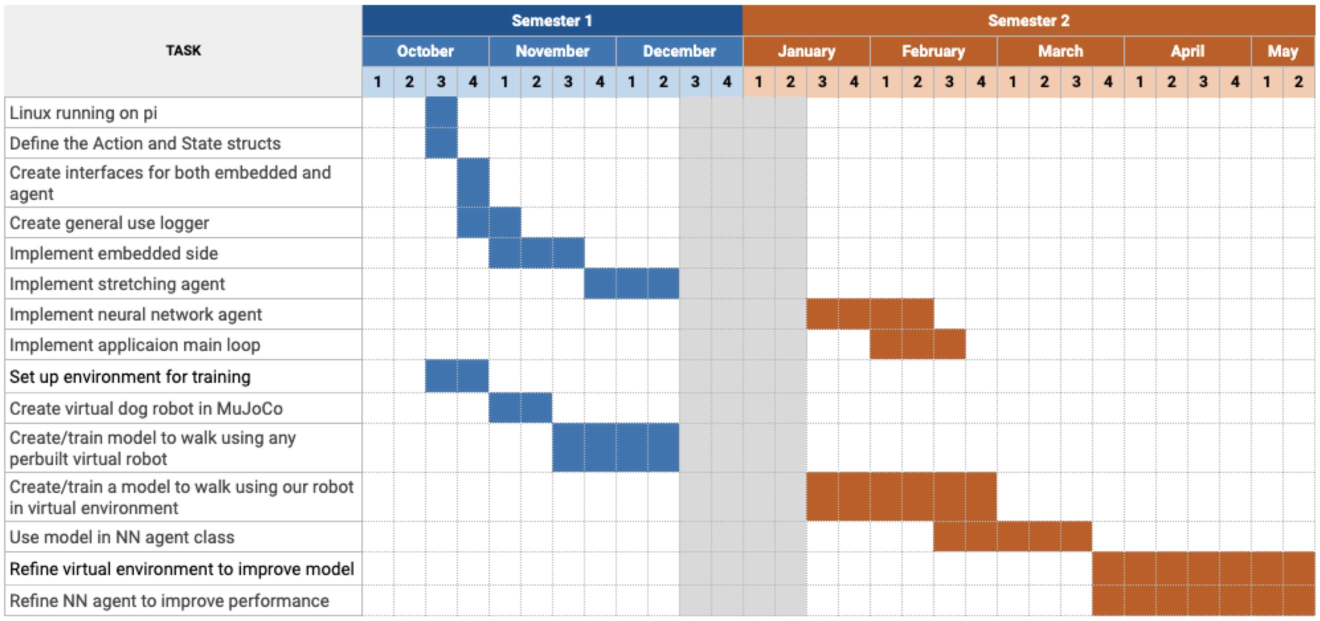
2.2 Task Decomposition

1. Get linux running on robot, run a basic hello world (C++)
2. Define the Action and State structs/arrays
    a. What values go in it and what they represent
    b. This is based on what is available from the robot
3. Create interfaces for both the embedded and agent (C++)
    a. This allows us to replace the agent easily without changing the application main loop
4. Create general use logger (C++)
5. Implement embedded side (C++)
6. Implement Stretching agent - IE proves that embedded side works and test robot joints if one is acting up (C++)
7. Implement NN agent (C++)
8. Implement application main loop - makes calls to interfaces, not to specific implementation (C++)
9. Set up environment for training (Py)
10. Create virtual dog robot in MuJoCo (Py)
11. Create + train a model to walk using any prebuilt virtual robot- proof of concept for training in virtual env using a prebuilt virtual robot such as the spider (Py)
12. Create + train a model to walk using our robot in virtual environment (Py)
13. Use model in NN agent class - actually trying the model in real life (C++ and Py)
14. Refine virtual environment to improve model as needed (Py)
15. Refine NN agent (C++) as needed - IE possibly modify the actions to account for friction or something

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Run basic “hello world” or similar program on the robot. (milestone). Evaluation: will be considered complete when the program runs successfully and as expected with no bugs.
- Completion of general use logger (milestone). Will need to distinguish between different types of information/error types (evaluation criteria). Will need to be easy to navigate for team members to easily find relevant data needed for debugging.
- Completing creation of virtual dog robot (milestone). Will need to accurately represent the physical model, including range of motion and dimensions (metrics)
- Completing the training environment (milestone). This will need to include different expected options that our robot will navigate through, will require us to sit down and decide what are appropriate additions to place in training (metric).
- Train a NN capable of walking in the virtual environment (milestone)
- Use the NN trained in the virtual environment on the physical robot (milestone)

2.4 Project Timeline/Schedule



TASK	Semester 1												Semester 2															
	October				November				December				January				February				March				April				May	
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Linux running on pi			🟦								░	░	░	░																
Define the Action and State structs			🟦								░	░	░	░																
Create interfaces for both embedded and agent				🟦							░	░	░	░																
Create general use logger				🟦	🟦						░	░	░	░																
Implement embedded side					🟦	🟦	🟦				░	░	░	░																
Implement stretching agent								🟦	🟦	🟦	░	░	░	░																
Implement neural network agent											░	░	░	░	🟧	🟧	🟧	🟧												
Implement applicaion main loop											░	░	░	░			🟧	🟧	🟧											
Set up environment for training			🟦	🟦							░	░	░	░																
Create virtual dog robot in MuJoCo					🟦	🟦					░	░	░	░																
Create/train model to walk using any perbuilt virtual robot							🟦	🟦	🟦	🟦	░	░	░	░																
Create/train a model to walk using our robot in virtual environment											░	░	░	░	🟧	🟧	🟧	🟧	🟧	🟧										
Use model in NN agent class											░	░	░	░					🟧	🟧	🟧	🟧	🟧							
Refine virtual environment to improve model											░	░	░	░										🟧	🟧	🟧	🟧	🟧	🟧	🟧
Refine NN agent to improve performance											░	░	░	░										🟧	🟧	🟧	🟧	🟧	🟧	🟧

This schedule allows us to divide our tasks into sprints typically lasting two weeks. Because we will be training virtually, much of the training can be done in parallel while work on the robot is being done. We will divide our team into one group for starting the virtual training process and another group for preparing to deploy our model on the robot. Towards the end of the second semester, we will all be able to help refine the ML model. There may need to be some training on the robot itself if the virtual training fails to accurately simulate the real world.

2.5 Risks and Risk Management/Mitigation

Task	Risk	Risk Probability	Risk Mitigation
Get linux running on robot, run a basic hello world (C++)	Robot delivery is delayed	0.2	
Define the Action and State structs/arrays	Incorrect action and state structs/arrays defined	0.2	
Create interfaces for both the embedded and agent (C++)	Implementation of interfaces takes longer than expected or doesn’t work as expected when using to easily replacing the agent	0.1	
Create general use logger (C++)	Logger does not work as expected	0.1	
Implement embedded side (C++)	Inexperience of team with implementing embedded side especially with ML	0.5	To mitigate this risk, we will seek external resources that have implemented something similar for assistance. We will also account for the lack of experience in our project timeline by allowing for some flexibility with this task if it is to take longer.
Implement Stretching agent	Trouble defining what actions we should specify	0.2	
Implement NN agent (C++)	Inexperience with tensor flow and loading a model	0.4	
Implement application main loop	More complex than we are predicting	0.3	

Set up environment for training (Py)	Environment is harder to set up then expected	0.8	Spend more time on it as needed
Create virtual dog robot in MuJoCo (Py)	We can’t get the data values we want	0.3	Figure out a way to convert the available data values to the ones we want by adding additional broken down tasks if necessary
Create + train a model to walk using any prebuilt virtual robot- proof of concept for training in virtual env using a prebuilt virtual robot such as the spider (Py)	Pre-built virtual robot is significantly different than our robot	0.5	Research pre-built robot thoroughly before selecting to make sure it will align well with how our robot works. If the options for a pre-built virtual robots are limited and none align well with our robot, then we can add additional tasks to our project plan to convert the pre-built implementation to our robot.
Create + train a model to walk using our robot in virtual environment (Py)	This tasks could be more challenging than we initially expected, could take more time than we initially plan for	0.5	In order to reduce this risk, we should allocate some extra time for this task when creating our project timeline
Use model in NN agent class	The model set up in the virtual world does not align well with the physical world, causing the completion of this task to take longer	0.7	It is likely that some modification will be needed when transitioning from virtual to physical environment, therefore, we should account for this upfront in our project plan and allocate enough time for refinement of the model.
Refine virtual environment to improve model as needed (Py)	More refinement is needed than planned for, we are unable to refine to where we need to be	0.8	To mitigate this risk, we are planning to allocate a significant amount of time towards this part of the project. We also will utilize our resources to become as educated as we can about our application, NN’s, and ML in order to reduce exposure to this risk.

Refine NN agent (C++) as needed	More refinement is needed than planned for, we are unable to refine to where we need to be	0.7	To mitigate this risk, we are planning to allocate a significant amount of time towards this part of the project. We also will utilize our resources to become as educated as we can about our application, NN's, and ML in order to reduce exposure to this risk.

2.6 Personnel Effort Requirements

Task	Person-hours	Explanation
Interface robot with Linux	**2h**	**It's designed for it, should be easy**
Create interfaces for both the embedded and agent	**2h**	**These should be easy to write as we have already discussed what they will look like, though its not set in stone**
Create general use logger	**1h**	**This is just a convenience class for logging, should be some basic file management**
Implement embedded side	**30h?**	**We are unsure what this will look like so we are allotting a significant amount of time to figure it out**
Implement Stretching agent	**4h**	**Just has to implement a few functions, can be a series of hard-coded actions**
Implement NN agent	**8h**	**This is more complicated, we need to read the docs for tensorflow to figure out how to load and use a model in C++. Note this does not include training the model, only loading it and getting an output from an input. We also will not be able to test it until we have a model that can be loaded.**

Implement application main loop	**4h**	**Well abstracted, this should only be a few lines of code. May require some additional arg parsing.**
Set up environment for training	**10h**	**Setting up envs in Python is notoriously a pain**
Create virtual dog robot in MuJoCo	**4h**	**This might require a lot of file editing and trial and error**
Create + train a model to walk using any prebuilt virtual robot	**15h**	**Figuring out how to properly train a model in MuJoCo will be a lot of learning and experimenting**
Create + train a model to walk using our robot in virtual environment	**15h**	**Once we have a virtual model of the dog, and we have completed the above step, training for the dog should be doable. Training itself may take some time**
Use model in NN agent class	**1h**	**If we have the agent class done, and the model done, we just need to load the model into the agent class. Should just be file transfer**
Refine virtual environment to improve model as needed	**TBD**	
Refine NN agent	**TBD**	

2.7 Other Resource Requirements

- OpenAI Gym: toolkit for training an agent using reinforcement learning
- MuJoCo: physics engine for building a virtual representation of our robot to train with OpenAI Gym
- Tensorflow: machine learning framework used to design our models and deploy them onto our robot
- Petoi Bittle: robot dog to deploy machine learning on.
- Raspberry Pi / microcontroller: Run ML model and interface with Petoi Bittle via I2C.
- A Linux machine for training