ELECTRICAL AND COMPUTER ENGINEERING
COLORADO STATE UNIVERSITY

DTVC Document

Rambots

Electrical Engineers:

Jared Davis, Ritwik Vadapally, Craig Olson, Oscar Coronado Rosales

Computer Engineers:

Kristopher Alquist, Devin Pohl

Junior Outreach Members:

Alex Kolodzik, Gwyn Tari

Advisor:

Olivera Notaros

Industry Advisors:

David Farrell

Ian Bernstein

EIR:

Eli Scott

Summary

The ECE Outreach team is proposing to design and build a Robotic Ram controlled by a Raspberry Pi 4B. This robotic ram will include cameras, custom actuators, and almost entirely 3D printed parts. The robot will be based off of two existing open-source robots; we aim to massively improve and extend functionality. The purpose of the project is to create a low cost, fast set-up time, and open source robotics platform to attract middle and high school students towards Electrical and Computer engineering as well as general robotics. The physical robot can be used to express electrical engineering concepts while the software can be used to express and show off the computer engineering field. The entire robot as a whole will give students a better idea of what robotics is like. Students of all education levels will be able to take a detailed look into our open source resources, referencing our work to create their own design to fit their goals.

Why This Project is Important

With the proliferation of robotic interest being seen today there are a lot of problems for those who seek to participate. The biggest being affordable quality platforms available for a wide variety of projects and disciplines. What this project aims to achieve is to make a high quality, affordable, and open-source robotics platform with the goal to spark interest in STEMM and promote the use of robotics in any discipline.

Objectives

This project has many objectives ranging in various categories and importance.

Appearance

Although the exact appearance has yet to be solidified, it was determined to be the size of a large canine: approximately 1 meter long by 0.5 meters high and weighing no more than 27.5 kilograms. The outward appearance will use curved edges with a chassis hiding all or most electrical components and wires. In addition, at the front of the robot there will be a ram head inspired by Colorado State University's Cam the Ram.

Functionality

The main objective is to have full quadrupedal motion. This entails, at the minimum, walking at the same pace as a canine of its size. Higher degrees of articulation such as running, climbing, and traversing various terrain are objectives that build off of each other as progress is made.

The other functionalities will be regarding its computer vision capabilities. This will primarily be detecting common objects with the ability to make selected objects motivate the RAMbot to carry out certain actions. Later, more sophisticated algorithms will be implemented such as facial recognition, and crowd detection.

Philosophy

The goal behind the creation of the RAMbot is to provide a low-cost, high-quality, open-source alternative to leading computing platforms in order to spark more interest in engineering and STEMM as a whole. The objective is to make every aspect of this project as transparent as possible. This includes a full parts list, all files regarding 3D printed materials, well documented code, and, if necessary, all custom PCBs. This may also include video or written tutorials in the construction of the RAMbot.

Design Specification/ Requirements

- **Hardware:**
  - Size(tentative): length - 39'', Height - 19'', Width - 15''
  - Weight - 60 pounds
  - Build Material: PLA, metal
  - Sensors
    - Camera - Facial Detection
    - Heat sensors - To differentiate between humans and objects
    - Gyroscope - To make sure the Bot balances itself
  - Safety
    - Curved edges
    - Controller - For manual override and to turn on/off
    - Electric sealing - To prevent getting shocked
  - Arduino - To act as a body controller
  - RaspberryPi - General connector and Machine Learning base
  - ESP32 - Real-time IO offloading from the Raspberry Pi
- **Requirements/Specifications**
  - Low Cost-plastic -- attempt to use old PLA from the Senior Design lab

- Able to walk and autonomous movement
- Uses the data from the sensors to detect humans, objects, hurdles etc;
- Open source for future generations of the project
- Able to take it to Football/ Volleyball games and High Schools for Outreach
- (Tentative) Sends notifications about errors
- Power
    - Battery variety of li-pos adjusted to motor need

Timeline

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16
Website	Devin	Devin	Devin											Devin	Devin	Devin
Budget	Jared Ritwik Oscar	Jared Ritwik Oscar	Jared Ritwik Oscar													
Project Plan	Jared Ritwik Oscar	Jared Ritwik Oscar	Jared Ritwik Oscar	All Team			Jared Ritwik Oscar	Jared Ritwik Oscar	All Team					Jared Ritwik Oscar	Jared Ritwik Oscar	All Team
FEMA			Craig													
3D Printing								Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig
Power requirements				Craig	Craig											
Electronics Choices						Jared	All team	All team								
Embedded Systems																
Purchasing			All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team	
Robot Build												Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik
Wiring												Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik
Programming					Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris	Devin	Devin	Devin	Devin	Devin
Testing												Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris
Machine Learning					Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik
DTVC Document										All team						

Figure 1: Timeline

Not Started	
In Progress	
Finished	Finished but will need revision
Revision needed	
Revision in Progress	

Figure 2: Timeline Color Meaning

Testing Procedures:

- **Software Tests:**
  - **HITL:** Due to significant constraints in obtaining parts, software development is progressing much faster than hardware development. In order to properly progress, we need a solution to test motors and sensors that we currently do not have.

    The solution is to use a *Hardware in the Loop* configuration. This is a testing configuration where a production microcontroller is hooked up to another system instead of real devices. This other system then simulates sensor and other input data, and validates output data generated by the microcontroller. This type of configuration will allow the testing of overall functionality without having parts present. Additionally, due to the offloading concept, the production microcontroller is not burdened with the processing overhead of validation and can run at full throttle. Furthermore, should delays be extended, we can spend time increasing the scope of emulation and simulation of the robot for increased validation.

    Specific areas for which this configuration will test include stress testing (can the microcontroller produce correct output when under heavy load), timing testing (can the microcontroller produce output fast enough), and unit testing of fundamental operations.

  - **Machine Learning:** Overall, the end goal is to have the RAMbot classify objects with an accuracy of 80%, and an inference time of 3 seconds or less. In addition, some object it is trained to detect can be labeled a priority, where upon detection it will attempt to move forward. The accuracy was determined to be the minimum accuracy in order to successfully demonstrate ML algorithms to an audience of introductory knowledge or less of ML applications. The inference time has the same motivation of being the bare minimum response time for good demonstration purposes. The exploration of ML algorithms will also be done on Waveshare Pippy, an open source bionic robot dog, that allows us to expand on more advanced ML algorithms concerning object detection on a smaller scale than the RAMbot. Since both are similar designs, the Waveshare Pippy will be used as a way to first test more complex algorithms before implementing and testing them on the RAMbot
  - **Test 1:** Training the ML Model

    · The machine learning algorithm will be trained on the Raspberry Pi with the Intel Neural Compute Stick II (INCS2) hardware acceleration. This will be done with 80% of images for training, and 20% of images for testing.

· *Pass*: When the training is complete, there are no fatal errors, warnings, or exceptions thrown that prohibited the full training of the ML algorithm.

· *Fail*: During the training processes, some error, warning, or exception that prohibited the full training of the ML algorithm. This may have been caused by a momentary power outage, or a hardware failure ranging from the INSC2 or Raspberry Pi board itself.

- **Test 2:** Testing the Model with Test Image-set

  · Using the trained model, use the test data set composed of 20% of the total images in the data set in order to obtain the accuracy of the model. This test does entail putting objects in front of a camera to see how well it classifies. This is meant to be a trivial test using images contained within the dataset that it didn't have access to when training.

  · *Pass*: The model has an accuracy of at least 80%, meaning it is able to successfully classify images well enough to handle real-time object detection using a camera and some objects close to those found in the data set.

  · *Fail*: The model has an accuracy less than 80%, this does not meet a base accuracy required to move onto the real-time object detection test with confidence. This could be either due to an insufficient ML model architecture, or poor data augmentation.

- **Test 3:** Testing the Model with Similar Objects in Real-Time

  · Using the trained model, present objects very similar to what is found in the dataset used to train/test the model. This test is a real-time object detection test, however using cherry-picked objects as close as one can manage to what is found in the dataset. For instance, if one image in the dataset is a water-bottle, this test would use a water-bottle looking very much like it. This will be done with 10 different objects that all belong to distinct classes of objects.

  · *Pass*: The model is able to classify 8 of the 10 objects successfully with an inference time of 3 seconds or less. This tells us the device not only can successfully classify a wide range of objects, but it also has a snappy response time.

  · *Fail*: The model was not able to classify the objects with an 80% accuracy, which can be caused by insufficient ML model architecture, or poor data augmentation.

· *Fail*: The model was not able to inference in 3 seconds or less. This could be due to an inefficient communication method between the camera and main board, or the ML model architecture is too slow for real-time inferencing.

- **Test 4:** Testing the Model with More Naturally Found Objects

  · Using the trained model, present objects that fall under the categories the model should know how to classify. This is a real time object detection test using objects more like what RAMbot would encounter in a more natural environment. For example, if the model is trained to see plastic water-bottles, it should be able to classify any plastic bottle within reasonable resemblance to one. i.e., an empty coke bottle. This test like the last will use 10 different distinct objects the model should be able to classify.

  · *Pass*: The model is able to classify 8 of the 10 objects successfully with an inferencing time of 3 seconds or less. This shows it is able to be demonstrated in a more natural environment where cherry-picked objects are not necessary for fair results.

  · *Fail*: The model is not able to classify 8 of the 10 objects. This shows the model is not generalizing the training dataset well enough to recognize objects that deviate too far from the training data. This could be caused by an insufficient ML model architecture, or poor data augmentation.

  · *Fail*: The model was not able to inference in 3 seconds or less. This could be due to an inefficient communication method between the camera and main board, or the ML model architecture is too slow for real-time inferencing.

- **Test 5:** Integration with RAMbot test

  · This test involves the integration of the machine learning model, and the full capabilities of the rest of the RAMbot. This test entails setting the RAMbot so that it recognizes a certain class of objects to be a priority, where when the ML algorithm classifies the priority, it will proceed to move forward. This test is meant to demonstrate the use of ML algorithms as a way to motivate action in the RAMbot, and successfully integrate the ML aspect with the rest of the design. This test will be performed using 10 objects it has previously successfully classified 80% of the time, where one will be given the label of priority. When the priority is shown, the RAMbot will attempt to move forward 100% if it recognizes it. In addition, the other 9 objects should not trigger movement.

  · *Pass*: If the object is recognized, RAMbot will proceed to try and move forward. This demonstrates not only the ML capabilities of RAMbot, but also the successful integration of the ML model with the rest of the design.

- *Fail*: The priority object is presented and does not trigger movement. This could happen because either it falls under the 20% of the time the ML algorithm fails to successfully classify the object. This could also fail as a result of a non-working connection between the Raspberry Pi the model is on, and the rest of the RAMbot implementation.

- *Fail*: A non-priority object is detected, categorized successfully, and then triggers movement. In this circumstance, the failure would occur as a result of a logic error in the code used to set an object as a priority, or an error in the code used to update the rest of RAMbot of a movement event.

- **Test 6:** Pippy Exploration Test

  - We will use Waveshare Pippy, an open-source bionic robot dog to test our machine learning models for object detection, body movement and body balance. The robot comes with an inbuilt camera to utilize for object detection and it also comes with a simulation software that we can use to simulate leg movement on different surfaces. Using this robot, we would be able to test our object detection by making the robot go around an object or go to a selected object, and we can use the robot on different surfaces to understand the performance and understand the readings achieved by the gyroscopes in the dog.

  - *Pass*: The dog is able to walk properly without falling or losing balance regularly, the dog detects different objects and walks to the desired object.

  - *Fail*: The dog is unable to walk and maintain its balance on uneven surfaces and keeps falling.

- **Hardware Test:**

  Correct Power Output

  With this robot having 12 motors and having to go through six motor controllers, correct power distribution is an absolute must and anything other than that will cause a failure in any ability to control or even balance the robot. The plan is to have two batteries with one strictly for powering the motors to cause less issues just in case there happens to be a large spike, where the motor controllers have a protocol for this while the raspberry pi without a proper breaker could be damaged. Having the separate battery is also for convenience though as well.

Motor Calibration

A very important part to the hardware that will most likely need to be run before even just standing up is calibration to all of the motors. This has to be done so that the dog does not destroy its own joints or perhaps damage anything around it. This is common for robots with only a certain degree of motion where the precision is an absolute must.

Test 1: Battery Distribution

On the final design of the robot we are planning on implementing live voltage readings through seven segment displays to allow the user to view each of the batteries readings, brain and motors, to confirm correct voltage.

The test would be seeing the response to a single motor driver to see the reaction to its performance, or even functionality. This could simply not work due to the motor controller's voltage regulator not being in the allowed voltage range or trying to power with very little current which will damage the battery. Understanding the reaction will help catch and prevent any future battery issues in the future.

Pass: A low charged battery will display its low charge and will not attempt to function due to that.

Fail: A low charged battery will not display its low voltage or will attempt to function with a low charge causing possible damage to the robot or battery.

Test 2: Motor Calibration

In order to prevent the robot from ripping apart its own joints, tests need to be run on motors before implementation on the final design of calibration to confirm calibration causes no errors. The test will need to check the position using the encoder and find its max rotation without going over. The test will also need to include some what if scenarios where the motor appears to start out of its allowed range.

Pass: The calibration for each scenario goes as expected and does not ever go out of its range in each scenario. When out of range it will not attempt to move and display the error.

Fail: During calibration the motor goes over its allowable range even 1 degree or attempts to move when starting out of range.