Project Rabbit

Design Document

Team sdmay22-22

Client: Kris Kunze - KCL Engineering

Advisor: Mai Zheng

Team Members:

Kevin Scanlon ……... Computer Engineer, Rabbit Team

Ben Dunkerton …. Computer Engineer, Computer Vision Team, Communications

Darron Anderson ……...Software Engineer, Computer Vision Team

Darrshen Balachanthiran ……..Electrical Engineer, Computer Vision Team

Lars Lofquist ……... Computer Engineer, Rabbit Team

Marshall Boser ……. Computer Engineer, Rabbit Team, App Development

Sam Fuller ……. Computer Engineer, Computer Vision Team

Team Email: sdmay22-22@iastate.edu

Team Website: http://sdmay22-22.sd.ece.iastate.edu/

Executive Summary

Development Standards & Practices Used

Standard circuit, hardware, software practices used in this project:

C/C++ coding standards
Python coding standards
Circuit design standards
Electrical and Electronic safety standards

Engineering standards that apply/that were considered:

IEEE 802.15.1 - Bluetooth
IEEE 1679.1-2017 - IEEE Guide for the Characterization and Evaluation of Lithium-Based Batteries in Stationary Applications
IEEE P2941 - IEEE Draft Standard for Artificial Intelligence (AI) Model

Summary of Requirements

- The Rabbit should cost less than 1000 dollars
- The app shall be available for both iOS and Android devices
- The Rabbit shall have an emergency stop function
- When the Rabbit is in motion it shall accelerate and decelerate similarly to a real runner
- The Rabbit shall be able to guide itself around a standard track
- The Rabbit shall have the ability to survive an impact from a running athlete
- The app shall have options for distance per set, number of sets and time to complete each set
- The app shall allow the user to connect to multiple Rabbit units
- The Rabbit shall be able to work continuously for at least 1 hour
- The Rabbit shall be able to match the pace of a human runner
- The Rabbit shall be directed by an app and controlled directly by an onboard microcontroller system
- The app shall respond to input in less than 5 milliseconds

Applicable Courses from Iowa State University Curriculum

- ComS 227
- ComS 228
- ComS 311
- CprE 281
- CprE 288
- CprE 309
- CprE 488

- EE 201
- EE 230
- EE 475

New Skills/Knowledge acquired that was not taught in courses

- Writing code in:
    - JavaScript
    - React
    - React Native
    - Arduino
- Using the OpenCV library
- Reverse engineering PWM signals
- Connecting microcontroller systems over I2C and SPI buses
- Passing data over bluetooth low energy connections

Table of Contents

List of definitions:

- Lap: traveling one full track length.
- Pace: target speed an athlete is trying to achieve when they are running on the track.
- Lane Detection: Using computer vision to identify the left and right lane lines of a track lane.
- Split: Each section of a workout that is defined by a time and a distance, eg. 100 meters in 10.5 seconds.
- Set: one or more splits combined to make up a workout.

1. Team:

1.1. Team Members

- Ben Dunkerton
- Darron Anderson
- Darshen P Balachanthiran
- Kevin Scanlon
- Lars Lofquist
- Marshall Boser
- Sam Fuller

1.2. Required Skill Sets for Project:

- Logical Problem Analysis
- Embedded Systems
- App development (Front End and Back End)
- Testing/Debugging
- Circuit Integration
- C/Python/Java
- Machine Learning
- Data Analysis

1.3. Skill Sets Covered by Team:

- Ben - Embedded Systems, App development, Testing/Debugging, C/Java
  Darron - C/Python/Java, Embedded Systems
- Darshen - Python, Embedded Systems
- Kevin - App Development (Full Stack), C, Java, Embedded Systems
- Lars - Embedded Systems, Java, C, Python, App Development (Full Stack), Arduino
- Marshall - App Development (Full Stack), Java, Python, Embedded Systems
- Sam - Embedded Systems, Java, C, App Development (Full Stack)

1.4. Project Management Style Adopted by Team:

- For the first semester we will be implementing a waterfall project style where we will tackle one problem at a time, and in the second semester we may consider agile development for our mobile application.

1.5. Initial Project Management Roles:

- Initially we will as a group democratically assign roles each week and share the results in our weekly meeting, rinse and repeat.

2. Introduction

2.1. Problem Statement

- In workouts, track athletes are often required to repeatedly run specifically paced intervals. This can oftentimes be a difficult task based on which lane the athlete is in, how many intervals they need to run, how well they feel physically, etc. While there are automated systems that exist to assist with this, they cost over $10k to install, and require constant maintenance. This leaves a majority of grade school and collegiate track programs in the dust when it comes to training efficiency and efficacy. Our goal is to develop a viable low cost solution to workout pacing that will ultimately be marketed to school athletic programs.

2.2. Requirements & Constraints

- Functional Requirements for the Rabbit:
  - The Rabbit shall have an emergency stop function
  - When the Rabbit is in motion it shall accelerate and decelerate similarly to a real runner
  - The Rabbit shall be able to guide itself around a standard track
  - The Rabbit shall have the ability to survive an impact from a running athlete
- Functional requirements for the App
  - The app shall have options for distance per set, number of sets and time to complete each set
  - The app shall allow the user to connect to multiple Rabbit units
- Non Functional Requirements for the Rabbit
  - The Rabbit shall be able to work continuously for at least 1 hour
  - The Rabbit shall be able to match the pace of a human runner
  - The Rabbit shall be directed by an app and controlled directly by an onboard microcontroller system
- Non Functional Requirements for the App
  - The app shall respond to input in less than 5 milliseconds
- Constraints
  - The Rabbit should cost less than 1000 dollars
  - The app shall be available for both iOS and Android devices

2.3. Engineering Standards

- IEEE 802.15.1 - Bluetooth
  - We are planning on connecting our system from app to car via bluetooth
- IEEE 1679.1-2017 - IEEE Guide for the Characterization and Evaluation of Lithium-Based Batteries in Stationary Applications
  - We are planning on powering our system with lithium Ion Batteries

and want to have safe way to handle and dispose of the batteries if needed

- IEEE P2941 - IEEE Draft Standard for Artificial Intelligence (AI) Model Representation, Compression, Distribution and Management
  - We are planning on using computer vision to solve our problem of staying in the center of a track lane

2.4. Intended Users and Users

- Users
  - This product is designed to be used as a low cost training bot for various levels of track athletics from grade school up to a collegiate level. The application platform provided would be used by coaches and athletes to design training routines that the Rabbit would then autonomously lead.By using the Rabbit the coaches and athletes have complete control over precise pacing and can maximize athletic performance.
- Use Cases:
  - Multiple Rabbits are used to create separate training plans for long and short distance runners.
  - Multiple Rabbits are used to train athletes at different levels (Ex. varsity and JV) with different intensities.
  - A Rabbit is used to pace a runner so the effect of the training program is maximized (Ex. runners who are “feeling good” don’t overexert in the early sets and tire themselves out for the later sets).
  - A Rabbit is used to train runners on the most efficient pace to keep for different sections of a race/set.
  - The Rabbit mimics running against a real competitor and therefore gives the runner something tangible to pace themselves against.

3. Project Plan

3.1. Project Management/Tracking Procedures

- Hybrid agile/waterfall project management
  - Justification: Our project elements are very much in lock step. We need to get the car to drive before we can get it to steer. We need to get an app up and running before we can connect it to the car. We feel that a waterfall approach will fit our project well since we plan on working in order on the tasks that need to be completed
  - Project Tracking: Github

3.2. Task Decomposition

Task Name	Super Task	Requires
Rabbit		
Physical System	Rabbit	● Select Microcontrollers ● Configure MCs to connect to one another ● Configure the MC system to connect to the Rabbit ● Decode PWM signals to build a vocabulary to communicate with the Rabbit ● Create mount for additional components
Onboard control/interface	Rabbit	● Physical System ● Bluetooth link to app ● Starting mechanism(Sound, Bluetooth, etc.) ● Batterylife (car to app)
Computer Vision	Rabbit	● Physical System ● Onboard control/interface ● Determine input source ● Line processing model ● Figure out steering algorithm
App		
Rabbit control	App	● Physical System, onboard control/interface ● Calculations to determine speed to set the Rabbit ● BT connection to MC System
User interface	App	● UX Design ● Basic Parameters for workouts

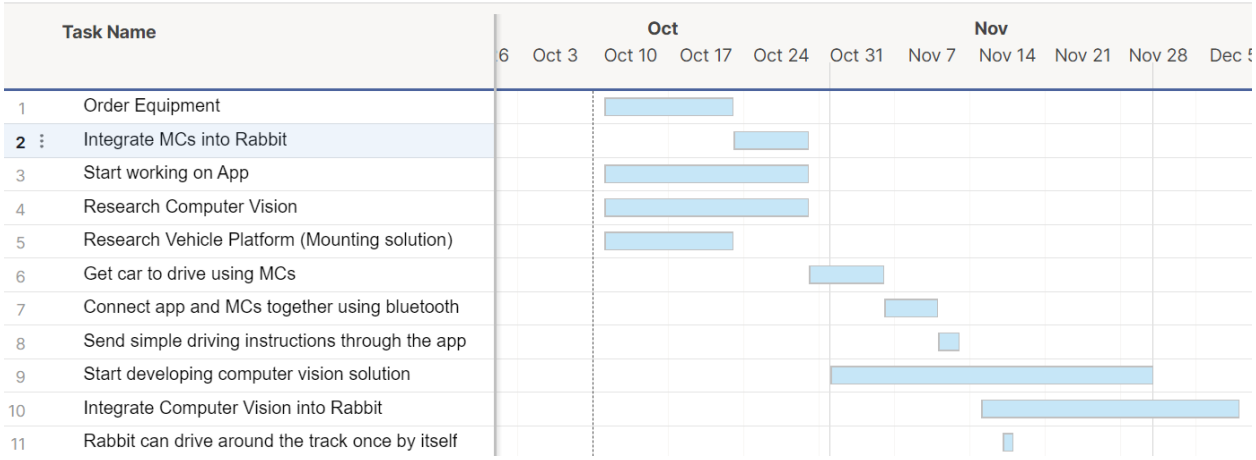
- Rabbit
  - Physical Systems
  - Computer Vision
  - Onboard control/interface
- App
  - Rabbit Control
  - User interface

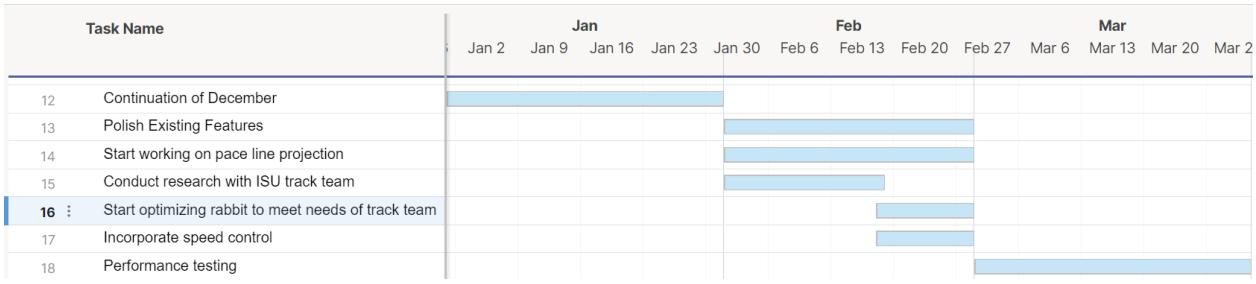
3.3. Project Proposed Milestones, Metrics, and Evaluation Criteria

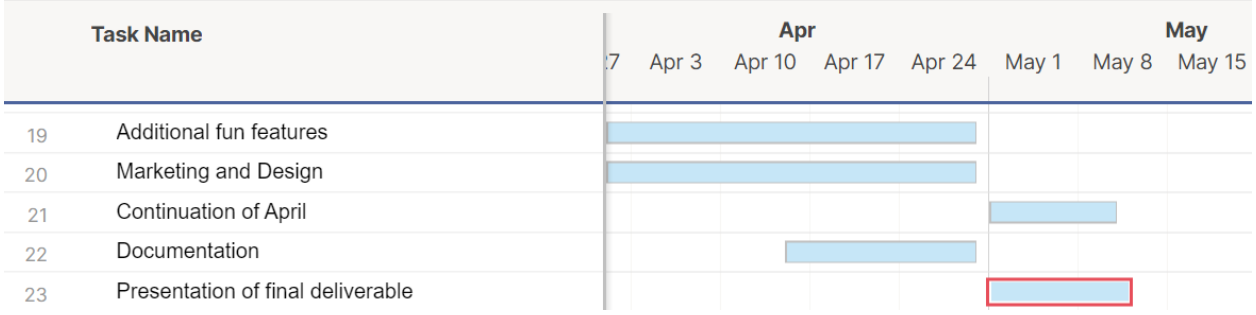
- Select/buy Micro Controllers
- Buy RC Car
- Select/buy sensors
- Integrate MCs into Rabbit
- Run simple driving commands from the MC (drive forward, *maybe* turn)
- Develop base level app
- Integrate app and Rabbit so app generated instructions can make the Rabbit move
- Develop Computer Vision
    - Computer Vision Program will take in input from camera and accurately register lines of a track
    - Computer Vision Program will take line data and extrapolate meaningful directional driving data to pass to the Adafruit Feather
    - Computer Vision Program will direct the Rabbit 1 full lap of a track

3.4. PROJECT TIMELINE/SCHEDULE

- October - Order project equipment/components, Integrate MCs into Rabbit, Start working on Basic UX for app, Research/start computer vision, research vehicle platform (mounting components)
- November - Get the car to drive using the MCs, connect app to MCs and send simple instructions
- December - Integrate Computer vision to MCs, Car can drive around the track by itself
- January - Continuation of December, catch up on any backlog work
- February - Polish existing features, projecting pace line, talk to Track Team
- March - Start optimizing, speed control, performance testing
- April - Fun features, Marketing Design
- May - Continuation of April, Documentation, Presentation


Task Name
Oct
Nov
6
Oct 3
Oct 10
Oct 17
Oct 24
Oct 31
Nov 7
Nov 14
Nov 21
Nov 28
Dec 5
1 Order Equipment
2 Integrate MCs into Rabbit
3 Start working on App
4 Research Computer Vision
5 Research Vehicle Platform (Mounting solution)
6 Get car to drive using MCs
7 Connect app and MCs together using bluetooth
8 Send simple driving instructions through the app
9 Start developing computer vision solution
10 Integrate Computer Vision into Rabbit
11 Rabbit can drive around the track once by itself


Task Name
Jan
Feb
Mar
Jan 2
Jan 9
Jan 16
Jan 23
Jan 30
Feb 6
Feb 13
Feb 20
Feb 27
Mar 6
Mar 13
Mar 20
Mar 2
12 Continuation of December
13 Polish Existing Features
14 Start working on pace line projection
15 Conduct research with ISU track team
16 Start optimizing rabbit to meet needs of track team
17 Incorporate speed control
18 Performance testing


Task Name
Apr
May
7
Apr 3
Apr 10
Apr 17
Apr 24
May 1
May 8
May 15
19 Additional fun features
20 Marketing and Design
21 Continuation of April
22 Documentation
23 Presentation of final deliverable

3.5. Risks And Risk Management/Mitigation

Risk	Likeliness	Impact	Mitigation Strategies
Feature development time exceeds projected timeline	0.9	Critical	We have built room for delays into our schedule
App to car communication - Takes too long - Out of Range	0.3	Low	Analog kill switch, program that kills the car when it detects a disconnect
Vehicle hardware failure - Delay to hardware development - Lack of platform to develop software on	0.2	Moderate	Purchase quality products from reputable suppliers Test product and fix any problems.
Computer Vision reading inaccurate -Potential of Rabbit crashing -Timeline gets pushed back	0.7	Moderate	Test computer vision before deploying to RC car

3.6. Personnel Effort Requirements

Task	Time Needed	Individual Contribution
Hardware Setup	12 hours	X3 people
MC Connection	5 hours	X2 people
MC Connect to Rabbit	5 hours	X2 people
Computer Vision	48 hours	X4 people
App(basic UX)	8 hours	X2 people
App integration to Rabbit	4 hours	X4 people
Computer Vision Integration to Rabbit	24 hours	X7 people
Additional Features	Infinity	All

3.7. Other Resource Requirements

Item	Cost	Link	Description
Track Car	$340	https://traxxas.com/products/models/electric/rustler-4x4?t=features	This RC car is capable of meeting human running speeds and provides a large enough platform for attaching microcontrollers and their necessary sensors.
Camera	$25	https://www.sparkfun.com/products/14028?src=raspberrypi	This is a high resolution camera that should be sufficient for detecting lane lines and can easily be connected to a dedicated camera bus on the Jetson Nano.
Car-Control Microcontroller	$30	https://www.adafruit.com/product/2995	The Feather M0 Bluefruit has a built-in bluetooth LE microprocessor for handling app-to-microcontroller communication. It also is faster than a standard Arduino Uno but can be easily programmed on the same platform.
Computer vision microcontroller	$60	https://www.sparkfun.com/products/17244	Jetson Nanos are power efficient microcomputers capable of computer vision processing with CUDA acceleration.

Sensors - GPS module - IMU - PWM Driver - Hall effect sensor	Price - $26 - $15 - $10 - $2	https://www.seeedstudio.com/Grove-GPS-Module.html https://www.adafruit.com/product/3463 https://www.adafruit.com/product/2928 https://www.adafruit.com/product/158	Various sensors for distance tracking, pwm signal generation, and acceleration tracking.
Vehicle battery & charger	$90	https://traxxas.com/products/parts/batteries/idpowercellbatteries/lipo/2869X-7600mah-74v-2S-25C	Self-explanatory, standard Traxxas battery and charger
Microcontroller batteries			We want to move away from microcontroller batteries, and instead use the same battery from the car to power all systems (currently not set up)

4. Design

4.1. Design Context

- Broader Context

Area	Description	Examples
Global, cultural, and social	Our project fits in the broad global and cultural scope of athletic performance. Currently only track athletes at the professional level have access to the track lighting system that aids in workout pacing. But once we deliver the Rabbit, access to high level training will be given to more athletes who are younger and will have more time to train in that space before they reach the age of most professional athletes. This could very well rebalance	A young athlete starts their high school track career with the Rabbit, and by the time they are a senior have trained with such accuracy that their performance rivals that of a seasoned veteran who has trained for twice as long without the aid of a pacing device.

	what the landscape of track competitions look like as more and more people will be able to increase their training efficacy and reach significant benchmark goals sooner in their career. (Might we unlock the sub 4 minute mile?)	
Public Health	The Rabbit may very well be useful for more than just an athletic workout. The Rabbit could also be used for personal workouts, or potentially for a rehabilitation program for someone who is relearning to walk or run. With the advent of telehealth, an automated pace car on a track could be a feasible way for a physical therapist to prescribe individualized workouts for patients who are retraining their legs to walk or run, and the Rabbit would be able to guide them through a session without the need for the PT to be there in person.	
Environmental	Since the Rabbit is a small product, we do not anticipate large impacts to the environment. However we will be using some form of battery to power the Rabbit, and potentially as the market for our device opens up, selling them in mass could increase the amount of batteries floating around out there which would in the long run lend	A College buys ten Rabbits, and after a few years throws away the old batteries instead of disposing of them in the proper manner.

	itself to a negative impact based on how well our users handle their batteries	
Economic	When a new product like ours enters the market there is always potential for positive economic growth. We anticipate that the benefits that come from utilizing a Rabbit in an athletic workout will be desirable at all levels of athletic ability from grade school, to collegiate, to professional. As such we would expect to not only sell a lot of Rabbits which would be good for the economy as it would create a solid flow of money, but it may also pique interest in the area of athletic performance in general and could generate even more market activity in the form of other workout enhancing solutions or parallel products that would all serve to increase athletic ability overall.	A college puts in an order for 10 Rabbits. After one or two seasons of use, the noticeable results drive other colleges to purchase and use the Rabbit which in turn creates higher demand that stimulates the market and leads to other innovative athletic training advancements.

- User Needs
  - Track athletes need a way to monitor their training so they can train as consistently as possible throughout the whole training routine.
  -
  - Track coaches need a way to tailor workouts to their athletes so that each athlete can excel and reach their full potential.
  - Grade school and collegiate athletic programs need a cheaper and more accessible solution than a track lighting system to allow their athletes to train and compete at a higher level.
- Prior Work/Solutions
  - An existing solution to this problem is a track lighting system. A very expensive and cumbersome solution that is only available to

top level athletic facilities. It provides pacing for only the inside lane and is not easily accessible for most grade school and collegiate athletic programs.
- Our solution is a better fit for the market we are looking at because we have a modular design that will work on any lane of the track, allow for more than one athlete to be doing a personalized workout simultaneously, and all at an affordable price with easy maintenance that a track lighting system can not provide.

■ Technical Complexity

Component	Subsystem	Applicable fields
Vehicle Hardware	Power,electrical	Safety standards, electrical equations
Line following	Computer vision, movement	Advanced computer vision and pathing
Real time communication from app to car(s) and car to app	Communications, App	Signal communication, parsing signals

4.2. Design Exploration

■ Design Decisions

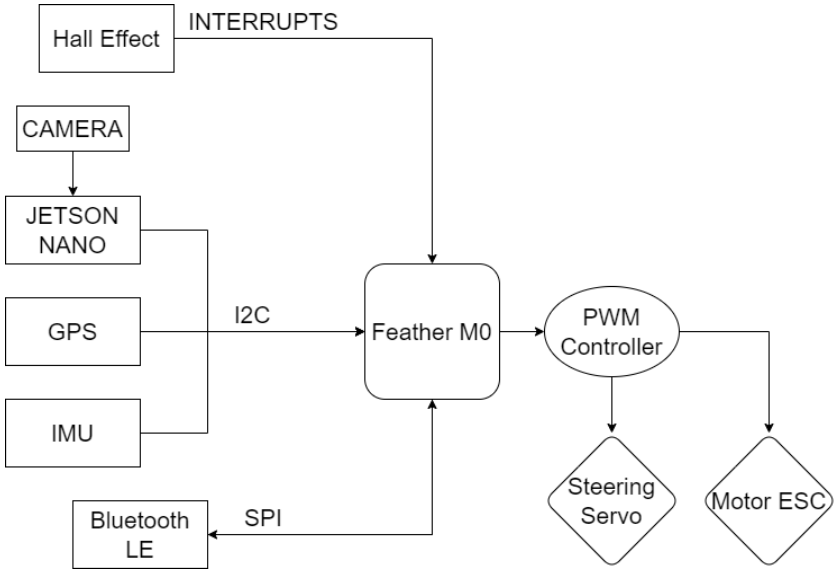
- What car should we use?
  - We decided to use the Traxxas Rustler, a car that is very balanced and has the performance power required of the Rabbit pace car.
- How will we handle line tracking?
  - We decided to implement the OpenCV library and utilize it to meet our specific requirements.
- How to interface with the car?
  - We ultimately went with a bluetooth low energy connection from the phone app to our Adafruit microcontroller.
- How to power the entire system
  - We still need to determine how exactly we will power the system, right now we want to use one main battery, but may need to include secondary batteries if individual components end up needing more power than expected
- How to mount all components necessary
  - We are currently considering creating a shroud that would replace the plastic topper of the RC car to allow us to custom mount our components

- Ideation
  - The design decision we spent a lot of time deciding and ideating on was the communication between the user and the car. The following ideas were ideated:
    1. Bluetooth
    2. Bluetooth Low Energy
    3. Exterior device to send RF signals
    4. Sound
    5. Cellular communication
    6. Server to host all the cars that the app can connect too
- Decision-Making Trade-Off

Criteria	Weighting	Bluetooth		Bluetooth LE		External RF		Sound		Cellular	
		Score	Total	Score	Total	Score	Total	Score	Total	Score	Total
Range	5	2	10	1	5	7	35	2	10	9	45
Cost of implemetation	2	7	14	8	16	5	10	9	18	1	2
Ease of implementation	3	4	12	7	21	2	6	3	9	5	15
Speed	4	6	24	6	24	3	12	7	28	1	4
Data bandwidth	1	7	7	6	6	3	3	1	3	4	4
			67		72		66		68		70

4.3. Proposed Design

- Our design is as follows: a purchased RC car serves as the base of the design attached to the car will be a platform for additional equipment we need such as a camera and boards for decision making this will be controlled via a smartphone app for both ios and android where a user can set distance and time settings and the car will automatically figure out speeds and acceleration. Once the user sends the ‘go’ command the car will execute these commands following the track through use of computer vision.
- Design Visual and Description



- The Adafruit Feather M0 accepts input from a variety of sensors, including speed data calculated from interrupts caused by a hall effect sensor, acceleration data from the IMU, and redundant speed data from GPS. The Jetson Nano is responsible for determining the position of the car within the lane, and then sends this corrective data to the Feather. Using sensor inputs and based on the workout plan delivered from the app, the Feather will then control the steering and speed outputs through a PWM module using a PID model. By comparing sensor data to the desired goals, a corrective error model can be generated continuously as the car makes its way around the track.

- Functionality
  - For an average user they will open the app, set up a connection to the car on their phone then set the distance, time and number of sets. After this is done they will ensure the car is in an appropriate position then once the runner is ready the user will hit the start button at which point the car accelerates in a manner similar to an average runner and follows the track with the runner following behind using the car as a guide for distance. Once the set is done the car shall come to a stop and be able to be retrieved and reset.
- Areas of Concern and Development
  - Computer vision is an incredibly complex topic and perfecting the system for our use presents unknown challenges and feasibility we plan to solve this by finding established research and ustalizing tutorials

4.4. Technology Considerations

- Using the Jetson Nano:
  - Pros:
    - We will be able to have high accuracy and efficiency with making driving decisions.
    - We will be able to quickly and easily make changes to the code if need be
    - We have the option to add other features if necessary
  - Cons:
    - The Jetson Nano needs a large amount of power and we may run into issues with powering the entire Rabbit off of one battery.
    - The Nano is also very large and will pose issues when it comes time to mount it onto the Rabbit body.
  - Solutions:

        - A solution for powering could simply be having a separate battery for the Nano.
        - We also plan on creating a shroud to hold all of the components and keep them secure during operation.
- Using Opencv:
    - Pros:
        - We can use the prebuilt libraries to detect lines and implement tools that already exist.
    - Cons:
        - We may spend time wrestling with getting Opencv to work properly and we do need a higher level of processing power to analyze the data.
    - Solutions:
        - Go back to implementing Opencv in python instead of cpp alleviate some of the more complicated aspects of the libraries.
- Using the Traxxas:
    - Pros:
        - High Durability
        - Open source design and easy to incorporate microcontroller functionality
    - Cons:
        - Original gear ratio was too low
        - Unsure of battery life
    - Solutions:
        - We were able to upgrade our gear ratio
        - Buy multiple batteries so we can swap them out if needed

- Coding our app in React Native
    - Pros:
        - App is portable to both android and iOS
    - Cons:
        - React hasn't been taught to any of us in our classe
- Adafruit Feather M0 Bluetooth
    - Pros:
        - Capable microcontroller with built in Bluetooth LE
        - Existing libraries with examples
        - Stackable additional modules
    - Cons:
        - Open source requires making code open source
        - No software UART support
        - Libraries often lack thorough documentation

4.5. Design Analysis

- While we have not fully completed our full design yet, it is coming together very well. Specifically our microcontroller system to control the car has operated and performed exactly as we anticipated it to. We have succeeded in establishing a high fidelity connection via bluetooth from a phone to the adafruit feather which we can sync to the pwm signals of the Traxxas car allowing us complete control over the car via our app. We have been able to send a list of commands as well as live driving instructions to the car. Our computer vision design has gone slower than we were hoping, but not slower than expected. We mainly have had blocker issues with getting Opencv properly installed on the Jetson nano. We recently cleared these issues up and plan on continuing forward with our original design implementation. The good news is that our design idea holds up to what we already have implemented with the RC car. We expect that once we have driving instructions generated from the Jetson Nano based on live video data, the driving instructions will successfully be followed by the RC car based on the testing we have already done through driving the car via bluetooth.

4.6. Design Plan

- Currently our design plan will continue forward without much modification. We will work out a way to interpret lines viewed by the Jetson Nano camera into meaningful driving instructions and then easily incorporate the Nano into our system flow by feeding its output to the adafruit feather input. Our overall progress will be governed by the success of the computer vision piece and as such we will plan on working through computer vision as much as we can over break so that when our spring semester starts we will give ourselves the best chance possible to start off on an even footing with a system that is ready to be fully integrated. Something that can help speed up our computer vision development is recording a video of a camera moving around a full lap of a track. We can then take this test data and use it as dummy input for our computer vision program. That way as we develop our solution for extrapolating driving instructions from the line data we will be able to retest and verify the success of our code over this data instead of needing to run the full Rabbit over and over again. This will greatly increase the speed of testing and overall development.

5. Testing

5.1. Unit Testing

ID	Units	Description	Tools
U.1	Car controller	Ensure that the car consistently operates	Oscilloscope

		using the same PWM signals.	
U.2.1	Car	Test to make sure the car can maintain a consistent speed set by our Microcontrollers	Speed gun
U.2.2	On car speed tracking	Verify our on board measurement system matches externally measured speed	Speed gun, hall effect sensor or gps
U.2.3	Car Battery	Test how long the car battery lasts	Oscilloscope
U.3.1	Computer vision	Verify that our computer vision solution recognizes only the lines we want	Camera, Jetson-nano
U.3.2		Verify that our computer vision solution make accurate decisions based on information passed in	Camera, Jetson-nano
U.4	App	Test that our app can calculate correct pace speed for given information	Junit/Mockito
U.5	MicroControllers	Ensure microcontrollers work	Arduino IDE

5.2. Interface Testing

ID	Name	Description	Tools
IF.1	MC	Test that the MC system can send driving instructions to car	Mock API
IF.2	App	Verify that the app can	Adafruit Feather

		send out valid instructions through bt low energy	
IF.3	Computer vision	Ensure that the Computer vision system can send driving decisions to the car	Mock video of track driving footage, Junit testing/mockito

5.3. Integration Testing

ID	Name	Description	Tools
IG.1	MC to Car	Test that the MC system can send driving instructions to car and the car follows them correctly	Arduino IDE
IG.2	Computer vision to MC	Ensure that the Computer vision system can send driving decisions to the MC system that forwards them to the car	Mock video of track driving footage, Junit testing/mockito Arduino IDE
IG.3	App to MC/MC to App	Ensure that the application can send valid instructions for the MC to process and pass on to the car and that the MC can send back valid information about battery life and speed	React Native, Junit, Adafruit Feather

5.4. System Testing

ID	Name	Description	Tools	Reference
S.1	Autonomous Driving	The car can drive at a constant speed around a track with no external	Car, feather, app (controller)	U.3.1, U.3.2, U2.2, U2.1, IG.1, IG.2

		assistance.		
S.2	App to Car	Testing that we can send information from the app to the car, and the car drives as expected	App, feather, car	U.4, IF.2, IG.3
S.3	Rabbit	Send a full workout program to the Rabbit and verify it completes the workout without leaving the track	App, track, CV, car, etc. (all aspects needed)	S.2, S.1

5.5. Regression Testing

ID	Name	Description	Tools
R.1	Basic Car Driving	Test that we can send a basic drive forward instructions to the Car whenever we make an update to the app	Track, App, Rabbit System
R.2	Computer Vision Accuracy	As we are developing computer vision, ensure that it consistently is making accurate decisions	Mock video of track driving footage, Junit testing/mockito
R.2.1	Computer Vision Line accuracy	Ensure that the computer vision solution can accurately identify lines	Mock video of track driving footage, Junit testing/mockito
R.3	App Instructions	Ensure that the app consistently sends accurate workout data through BT low energy	Arduino IDE, React Native, Junit, Mockito

5.6. Acceptance Testing

Number	Area	FR/NFR/Constraint	Requirement	Test method
1.1.1	Rab	FR	estop	Start the Rabbit hit the estop
1.1.2	Rab	FR	Acceleration	Measure the performance of real runners using an accelerometer to find performance
1.1.3	Rab	FR	App control	Change options in the app and make sure the Rabbit changes its behavior
1.2.1	Rab	NFR	Battery life	Run the system for a hour going through routines
1.2.2	Rab	NFR	Match Pace	Measure the top speed of the Rabbit ensure it is above 27.77 miles per hour https://www.britannica.com/story/how-fast-is-the-worlds-fastest-human
1.3.1	Rab	Constraint	Cost	Put together a cost estimate of all parts and components and estimated expenses as well as a profit percentage and ensure it is less than 1000 dollars
2.1.1	App	FR	Options (NFR?)	Are the options available
2.2.1	App	NFR	Input Response	Measure the response time of various inputs and take the average
2.3.1	App	Constraint	App Availability	Is the app on both IOS and Android
3.1.1	Nav	FR	Lane	95% of tests have the

			following	robot stay in the lane

5.7. Results

- We have been able to run testing on the physical components of the system, but have not yet gotten to the point of testing our software as we still need to do the majority of the work for our computer vision program

6. Implementation

- Work with the track team to get data for how to model human acceleration.
- Complete the physical prototype of the Rabbit
  - Integrate the Jetson Nano with the microcontroller for lane detection.
  - Build a unique car shell to hold/protect all the components.
- Improve the autonomy of the Rabbit
  - Transfer our preliminary Python lane detection into C++ with GPU support in order to meet timing deadlines.
  - Expand the lane detection to include checks and balances for a more robust program. (See figure 8.4.2)
  - Output directions to the mircocontroller from lane detection.
  - Test across multiple tracks, indoor and outdoor, to fine tune image preprocessing.
- Enhance the App UX to support the full scale of the project.
  - Building a running routine.
  - Better remote control.
  - Switching between remote and autonomous control.

7. Professionalism

7.1. Areas of Responsibility

Area of Responsibility	Definition	NSPE Canon	IEEE (own interpretation)	Differences
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence	Perform services only in areas of their competence; Avoid deceptive acts	The ability to continuously learn and correctly implement knowledge gained through experience and education as we are qualified	NSPE Canon deals more with staying within areas of expertise while IEEE is more about making sure the work done is done at a level of excellence

Financial Responsibility	Deliver products and services of realizable value and at reasonable costs	Act for each employer or client as faithful agents or trustees.	Act ethically and do not compromise public service or show favor based on private payments	NSPE is about financial responsibility to the people who your working for while IEEE adds in taking into account the public
Communication Honesty	Report work truthfully, without deception, and understanda ble to stakeholders	Issue public statements only in an objective and truthful manner; Avoid deceptive acts	Be truthful and open with what we are doing so that the public can make informed decisions which will be in their best interest	NSPE doesn't include the consideration that the public can make their own opinion based on the facts
Health, Safety, Well-Being	Minimize risks to safety, health, and well-being of stakeholders	Hold paramount the safety, health, and welfare of the public	Work to prioritize the well being of customers and put their safety first above profits	Both are similar in nearly every way, with IEEE adding that safety should be prioritized over profit
Property Ownership	Respect property, ideas, and information of clients and others	Act for each employer or client as faithful agents or trustees	If we receive help from others, credit them properly, and if we make a mistake, publicly acknowledge it.	NSPE deals with honesty towards the employer/client while IEEE deals with properly crediting contributions. While different they both involve being honest about the project.
Sustainability	Protect environment and natural resources locally and globally		Seek to prioritize environmental safety and sustainability	NSPE does not have a Sustainability column.
Social Responsibility	Produce products and services that benefit society and communities	Conduct themselves honorably, responsibly, ethically, and lawfully so as	Seek to act ethically as we interact with clients, other engineers, and companies	NSPE deals with honor and responsibility and reputations. IEEE is about respecting those who may

		to enhance the honor, reputation, and usefulness of the profession		have differing backgrounds and upcoming.

7.2. PROJECT SPECIFIC PROFESSIONAL RESPONSIBILITY AREAS

Area of Responsibility	Performance	Reasoning
Work Competence	High	Since this task is relatively complicated it is important that we work efficiently and well together in order to get the Rabbit done on time
Financial Responsibility	Medium	We do have access to a decent amount of money thus it is important that we handle it responsibly
Communication Honesty	High	Since we will be working closely with the client it is important we keep him up to date and in the loop regarding the status of the project
Health, Safety, Well-Being	Low	Since this app is designed to assist peoples health and only involves a small rc car the impact to safety is reasonably low
Property Ownership	Medium`	Since we are developing a product for a client it is important that we keep the clients needs in mind throughout development
Sustainability	Low	The impact of the product will not be wide spread unless it becomes incredibly popular which is unlikely at this stage
Social Responsibility	Medium	The goal of this app is to improve training techniques of athletes if not handled well it could disproportionately affect higher income athletes so it is important we consider affordability in our process

7.3. Most Applicable Professional Responsibility Area

- The responsibility of Work Competence is important to our project and we have demonstrated a high level of proficiency thus far this semester in it. Because our project contains multiple systems and spans several disciplines from software to hardware we have all worked hard to build out

the various areas of our project and work together to ensure that the different areas will fit well together when it comes time to synthesize our system.

8. Closing Material

8.1. Discussion

- Since we are only half way through our project we don't have any solid results yet. But our progress on driving the car using bluetooth is very promising. It is a fantastic proof of concept showing that once we get computer vision working fully, this is a viable solution for providing accurate and accessible pacing technology to grade school and collegiate track programs. In addition to this we have been able to get the basics of lane detection so that given a picture of a track we can detect the lane the rabbit should be in.

8.2. Conclusion

- So far, we have only successfully implemented the remote control driving portion of our project. We are able to control the Rabbit using an app connected to the Adafruit Feather that sends PWM signals to the car itself. Our next goal is to implement and integrate a computer vision solution that will detect lane lines on a track and extrapolate meaningful driving instructions that will be fed to the car through the Adafruit Feather. The best plan of action we can take is to take some time over winter break to continue learning about and working on our initial implementation of the computer vision solution. Working over break will allow us to hit the ground running at the start of spring semester and set us up well for success. One major blocker issue we had was setting up the OpenCV libraries on our Jetson Nano. We had a lot of issues installing, building, and linking the libraries which took a significant amount of time to figure out. In the future, we could start earlier by setting up our resources and ensuring they are properly installed so we don't run into issues in the middle of a coding sprint.

8.3. References

Benoit Blanchon, "Efficient JSON serialization for embedded C++," *ArduinoJson*. [Online]. Available: https://arduinojson.org/. [Accessed: 06-Oct-2021].

L. Ada, "Adafruit Feather M0 Bluefruit Le," *Adafruit Learning System*. [Online]. Available: https://learn.adafruit.com/adafruit-feather-m0-bluefruit-le. [Accessed: 14-Sept-2021].

M. Hardwick, “Simple lane detection with OpenCv,” *Medium*, 22-Aug-2018. [Online]. Available: https://medium.com/@mrhwick/simple-lane-detection-with-opencv-bfeb6ae54ec0. [Accessed: 01-Nov-2021].

Mosh, “React native tutorial for beginners,” *YouTube*. [Online]. Available: https://www.youtube.com/watch?v=0-S5a0eXPoc. [Accessed: 28-Sep-2021].

Mosh, “JavaScript tutorial for beginners.” YouTube, [Online]. Available: https://www.youtube.com/watch?v=W6NZfCO5SIk. [Accessed: 17-Sep-2021].

R. Politiek, “Install opencv 4.5 on Jetson Nano - Q-engineering,” *Q*, 01-Dec-2021. [Online]. Available: https://qengineering.eu/install-opencv-4.5-on-jetson-nano.html. [Accessed: 06-Dec-2021].

8.4. Appendices

8.4.1 Team Contract

Team Name:
Rabbit [Racing Autonomous Bluetooth Bot Intended for Tracks] (Team 22)
Members:

- Ben Dunkerton
- Darron Anderson
- Darshen P Balachanthiran
- Kevin Scanlon
- Lars Lofquist
- Marshall Boser
- Sam Fuller

Procedures:
We are planning on having a weekly team meeting, with Discord as our main means of communication. We will make decisions democratically and at the end of every meeting, note any major updates or decisions made. We also plan on keeping a GitHub repo active as well as a wiki tracking our progress.

Participation Expectations:
We expect full participation, active communication, and regular attendance for meetings.
We expect a team member to give 24 hours notice when possible if they will not be able to make the team meeting.
We expect all team members to volunteer for various tasks.
We expect every team member to have a positive attitude and buy into our project.
We expect every team member to properly allocate time each week in order to complete any assigned task.

Leadership:

Currently we are working with a democratic approach to assigning roles in terms of weekly tasks and determining what items need to be completed.

Collaboration and Inclusion:
For the most part, all team members share skills, however Darron and Darrshen are our experts in Software and EE respectively. To include their specific skill sets, we will set them as leads on the specific parts of the project that would utilize their expertise. All team members will share thoughts and seek to hear thoughts so that as we are brainstorming, the best and most viable ideas will rise to the top.

Goal-Setting, Planning, and Execution:
Goals will be set on a week to week basis decided democratically

Team goals for semester:
We will brainstorm as a large group to set up overall goals for the semester, and after large goals are set, we will break them down into smaller and individual tasks.
To stay on task we will give weekly updates on project statuses and ensure to never assign a project task to only one person. This allows for increased accountability and throughput. We will also do code review style follow ups on tasks in order to confirm satisfactory results.

<u>Our current team goals for the semester are:</u>
Design the Rabbit
Get to a point where it can drive autonomously around the track
Start working on a base for our App

Consequences for Not Adhering to Team Contract:
If anyone is late to a meeting, they must be the test user for one test session of the Rabbit car.

If a team member is late to 2 meetings in a row, or 3 meetings in a month, we will have them re-evaluate their participation level and adjust their schedule and or work load for the project accordingly.

If there are any instances of server infractions (i.e. not completing work, disrespect to other members, or unprofessional behavior that represents our team poorly) we will first meet with the team member in question, if the behavior continues, bring this to our TA, and if necessary our professor.

****

a)
I participated in formulating the standards, roles, and procedures as stated in this contract.
b)
I understand that I am obligated to abide by these terms and conditions.
c)
I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) Ben Dunkerton
DATE: 9/17/21

2) Darron Anderson
DATE: 9/17/21

3) Darrshen Balachanthiran
DATE 9/17/2021

4) Kevin Scanlon
DATE 9/17/21

5) Marshall Boser
DATE 9/17/21

6) Samuel Fuller
DATE 9/17/21

7) Lars Lofquist
DATE 9/17/21

8.4.2 Lane Detection Diagram