Fetcher Car for Ball Sports

ECE 445 Design Document

Project Team #34

Louie Davila, Yinshuo Feng, Patrick Sarad

TA: Sainath Barbhai
Professor: Viktor Gruev

Contents

1 Introduction 3

1.1 Problem and Solution 3

1.2 Visual Aid 4

1.3 High-Level Requirements 4

2 Design 5

2.1 Block Diagram 5

2.2 Physical Design 5

2.3 Subsystem Descriptions 9

2.3.1 Power Subsystem 9

2.3.2 Control Subsystem 11

2.3.3 Sensor Subsystem 12

2.3.4 Fetching Subsystem 14

2.4 Software Design 15

2.5 Tolerance Analysis 16

3 Cost and Schedule 18

3.1 Cost Analysis 18

3.2 Schedule 19

4 Discussions of Safety and Ethics 20

1 - Introduction

1.1 - Problem and Solution

It is well known that athletes all around the world train for hours every day in order to hone their skills and become the best at their sport. Of course, no matter the training regime, athletes simply can't train all day and have to partition their time accordingly in order to maximize the gains they can make. As such, we can see here that time is a key factor when it comes to practicing. This is especially the case when people have to repeat a certain skill for hours in order to master it. Now, it is important to note that this isn't always the case. For instance, pro-athletes train for a living so they have more time to hone their skills. Younger athletes, on the other hand, often have an extremely tight schedule and have to fit in their training with school work or something similar. Our project aims to help these athletes in their training. More specifically, our project is designed to help athletes that train in ball sports.

In ball sports, many athletes can probably attest to both the annoyance and time-consumption of having to collect their ball after practicing a certain skill. For instance, in basketball, when practicing your shots, you may find that once you shoot all the balls that you had on standby, you have to go and pick them all up and bring them back to your initial training position. As another example we have tennis, where players have to go and pick up all of the tennis balls they used to practice their serves. The collection of these balls is a huge waste of time and, in many cases, takes more time than the actual act of practicing the skill. We intend to solve this problem by using a small, lightweight car with pincers that can watch for a ball from a safe position and fetch the ball when it leaves a designated region for practice. After fetching the ball, the car will return the ball to the user and will continue waiting for the next ball. This solution is the first of its kind in the sense that, with more time, manpower, and resources, the design could be expanded upon and refined for use in any ball-oriented sport.

1.2 - Visual aid

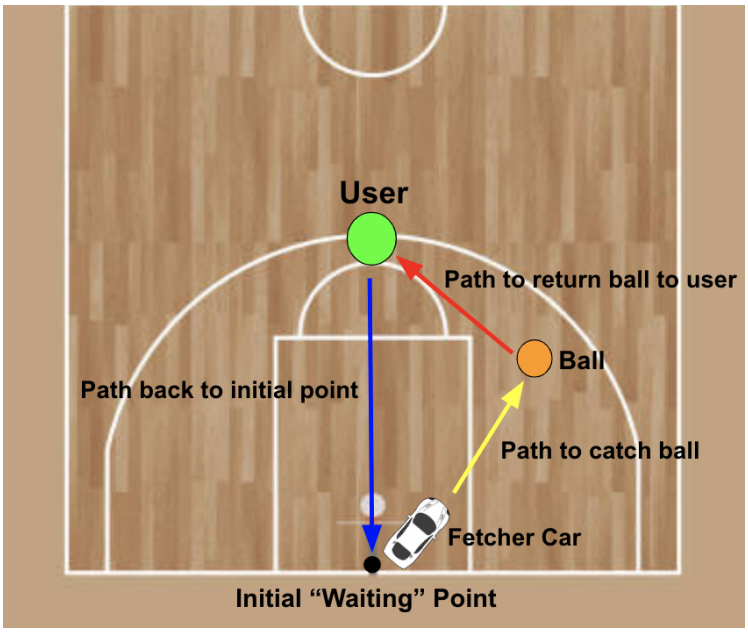

Figure 1: Process of Fetching a ball on a basketball court

Our project focuses on three main stages in order to successfully complete its task. The first stage involves the fetcher car searching for balls from a designated “waiting” point. Once a ball has been found, the car will proceed to follow the ball till it manages to catch it. This process is outlined by the yellow line in figure 1. Upon catching the ball, the car will begin to search for the user and will return the ball to them as soon as they are located. This is outlined in figure 1 with the red line. More specifically, the fetcher car will be located at the orange point and will follow the red path once the user is found. After this process is completed, the drone will follow the blue line (as shown in figure 1 again) till it reaches its “waiting” point and will once again begin the process from the beginning.

1.3 - High-level requirements

Our project will focus on the following high level requirements in order to solve the given problem in 1.1:

1. The fetcher car can locate the ball, the user, or the waiting location as necessary in the current context of the program within 10 seconds of beginning a search.
2. The fetcher car will be able to run with the wheels at maximum power output for at least 45 minutes before needing to recharge.
3. The fetcher car will be able to capture and return a ball to the user within 45 seconds of the ball becoming acquirable.

2 - Design

2.1 - Block diagram

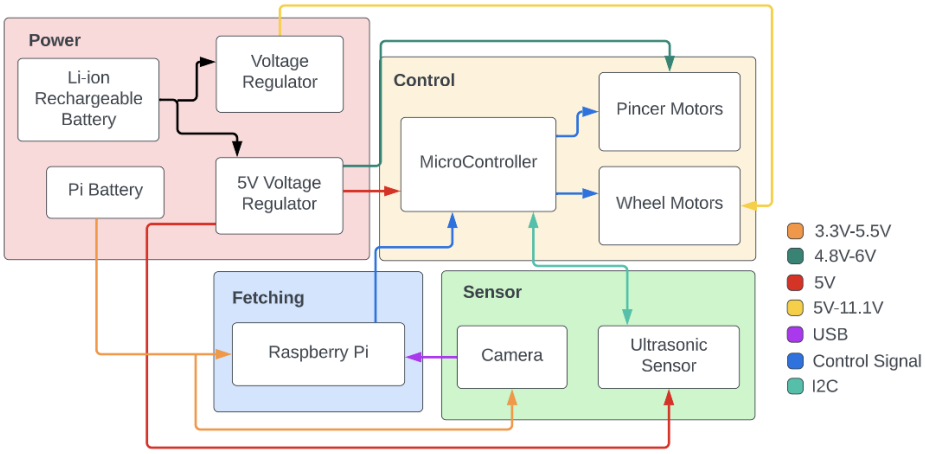

Figure 2: Fetcher Car Block Diagram

2.2 - Physical Design

The entirety of our project will be laid out onto a 4-wheel car chassis. This chassis contains 2 floors, each of which will be used to hold the corresponding components of each subsystem. Both levels have the same dimensions of 10.6 x 6.6 inches with a separation of 1.8 inches between them. Figure 3 displays the layout of the chassis with the corresponding dimensions.

Figure 3: Physical dimensions of car chassis

The first level of the chassis will primarily be used to hold the battery and the sensors. We will be using a rechargeable battery with the approximate dimensions of 2.8 x 2.8 x 2 inches near the back of the car. This is indicated by the red box in figure 4. In addition to this, we will have 2

motors powering pincers at the front of the chassis. The placements of these motors are indicated by the blue boxes in figure 4. These motors are 1.57 x 0.78 x 1.43 inches and won't fit between the levels of the chassis. As such, we will either keep them slightly in front of the chassis or we will adjust the top level of the chassis accordingly so that the motors can fit without issue. The size of these motors can be seen in figure 5 where the golden rod indicates the height of the second level of the chassis. The final components on this floor will be a pair of ultrasonic sensors as well as a camera. The camera will be placed between the 2 motors, as indicated by the green box. The ultrasonic sensors will fit beneath the first level so that they can be placed approximately level to the pincers. The relative positioning of these ultrasonic sensors can be seen in figure 5.

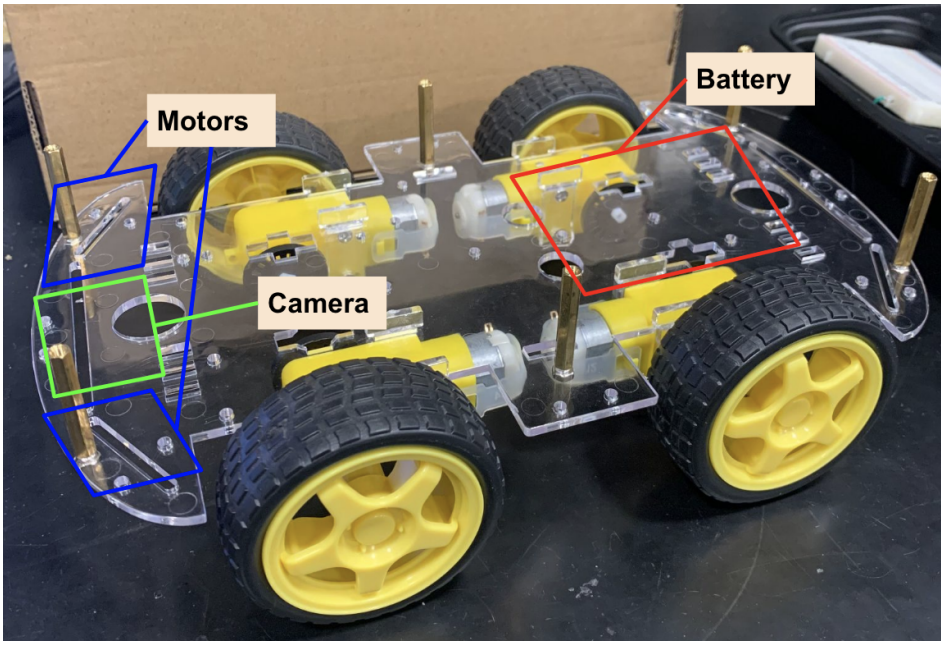

Figure 4: Component placements on first floor of chassis

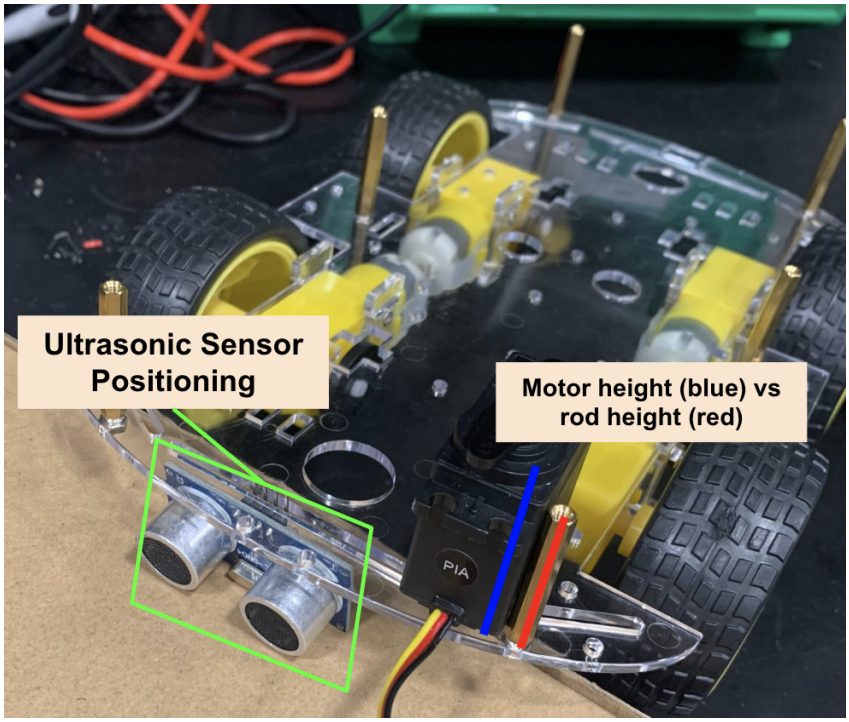

Figure 5: Positioning of ultrasonic sensor and pincer motor

The pincers need to be able to completely encompass a tennis ball and partially encompass balls of slightly bigger size. As such, the pincers need to be both spaced out at least 2.6 inches and need to be at least 2.6 inches in length in order to match the dimensions of a tennis ball. Of course, this would mean that we would be tightly holding the tennis ball which would add some friction working against the wheels of the car. The solution to this is to have the pincers both spaced out more and to have them be far longer than the ball to allow some space for the ball to move. We plan on spacing the pincers about 5 inches apart and we plan on them being about 5-6 inches in length. The pincers will be attached to the motors indicated by the blue boxes in figure 4. Figure 6 displays the pincers when opened in front of the chassis. Figure 7 displays how the pincers will enclose a ball.

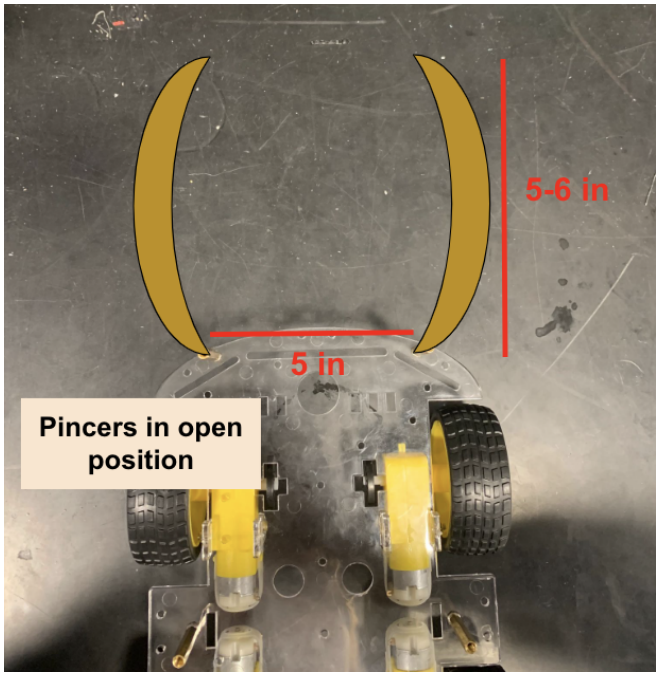

Figure 6: Pincer placement and dimensions

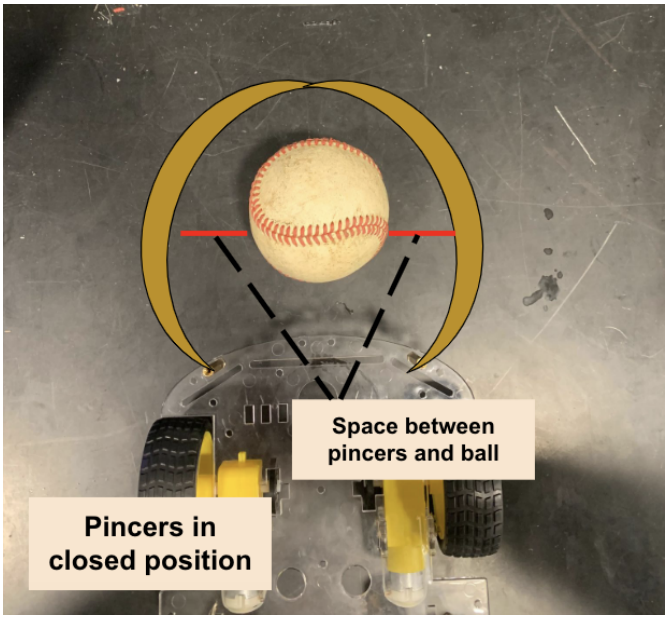

Figure 7: Pincers in closed position

Finally, on the second floor of the chassis we will place our pcb and raspberry pi. Due to various holes within the chassis, the exact positioning of the pcb and raspberry pi won't matter too much. These holes will allow us to place wires within the chassis without issue and hook up everything with ease. With that in mind, we will place the raspberry pi near the front of the chassis and place the pcb right behind it. Having the raspberry pi closer to the camera will make it easier to connect the 2 while having the pcb near the center will make the wiring of all the

motors more contained. Figure 8 shows the positioning of the 2 on the physical chassis, with the red box indicating the raspberry pi and the blue box indicating the pcb.

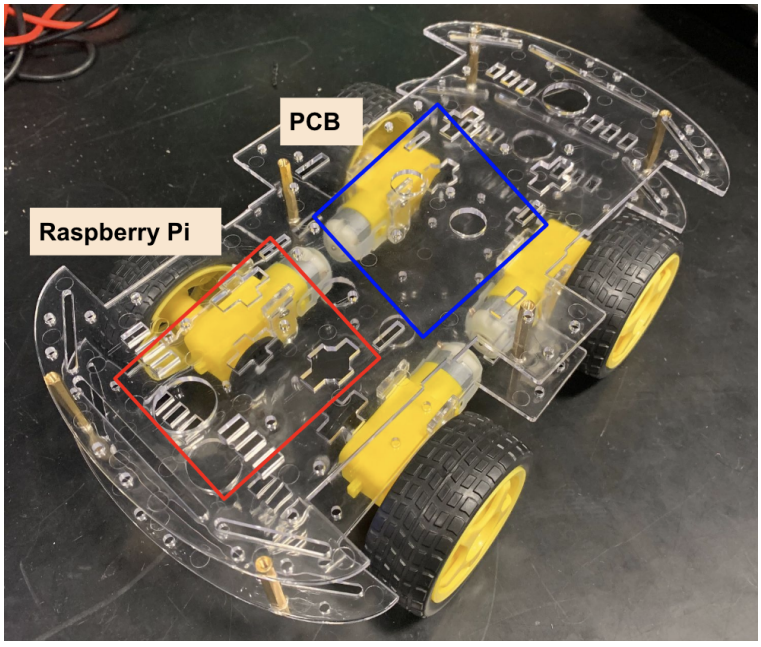

Figure 8: Component positions on the second floor of the chassis

2.3 - Subsystem Descriptions

2.3.1 Power subsystem

This block will be responsible for providing power to the PCB, motors, and raspberry pi. It will be responsible for ensuring that the design can move quickly enough to capture a ball within the time limit and remain active for at least an hour without the need for a recharge.
The motors on the right side of the car will be connected to each other in parallel, and the same will be done for the left side motors. The 2 pincer motors will also be connected to each other in parallel. Each pair of parallel motors will be controlled via its own designated H-bridge. The poles of each pair of motors will be connected to one of the 2 output pins on the H-bridge to allow for bidirectional current.

Notes on Equipment: All measurements relevant to the power subsystem will be taken using a multimeter (to measure voltage and current) and a scale (to measure the weight/mass of the car). A user-controlled, RC version of the control subsystem will be used to perform unit tests. This RC version will be hosted on the Raspberry Pi.

Requirements	**Verification**
This subsystem must provide a constant supply of 2A-4A at 3.3V-5.5V to the	Measure the current and voltage across the 5V and GND pins. Start running the fetching program (image processing and

Raspberry Pi for the duration of the time that the design is in use	communication with PCB). measure the voltage and current the same way as before. If the voltage and current are consistently within the expected range (given fully charged, working battery), then this requirement is satisfied.
It must also provide 1mA-5mA at 3V-5.6V to the ATmega328P microcontroller when idle and 9mA-30mA at 3V-5.6V when active	With the Pi off and the motors detached, check the voltage and current at the VCC and GND pins of the microcontroller. Connect the Pi and run the fetching program. Check the voltage and current for the microcontroller the same way as before. Connect the 3 pairs of motors and check the voltage and current the same way as before. If the voltage and current are consistently within the expected range (given fully charged, working battery), then this requirement is satisfied.
The ultrasonic sensors must be provided with a 10mA-20mA each at 4V-5.5V each so that they remain available for use at any time	With the Pi off and the motors detached, check the voltage and current at the VCC and GND pins of the 2 sensors. Connect the Pi and run the fetching program. Check the voltage and current for the sensors the same way as before. Connect the 3 pairs of motors and check the voltage and current the same way as before. If the voltage and current are consistently within the expected range (given fully charged, working battery), then this requirement is satisfied.
The pincer motors will be provided with 6mA-10mA (idle) and 160mA-185mA (active) at 4.8V-6V each so that they have enough power to move and capture the ball.	Provide power to the car. Prepare the multimeter to measure the voltage and current across the pincer motors, control the pincer motors with the manual controls, and record the current/voltage observed. If the measured current and voltage for each pair of motors is within its respective required range, then the requirements for the power subsystem are satisfied.
The car motors will use 0mA-70mA (idle) and 0.4A-1.5A (active) at 6V-11.1V each so that	Provide power to the car. Prepare the multimeter to measure the voltage and current across the left-side motors, control

they can carry all of the hardware needed for the design (including the ball).	the left-side motors with the manual controls, and record the current/voltage observed. Repeat with the right-side motors. If the measured current and voltage for each pair of motors is within its respective required range, then the requirements for the power subsystem are satisfied.

2.3.2 - Control Subsystem

This subsystem will be responsible for the transmission of all signals involved in the operation of the motors, and it is implemented on the microcontroller. The microcontroller will take input signals from the Pi and ultrasonic sensors, and the output signals will be to the ultrasonic sensors and/or any combination of the 3 H-bridges.

The absolute maximum value of the current at the microcontroller's I/O pins is 40mA, but our design does not have the ability or need to supply current at or above this value to any of the I/O pins. The inputs to the microcontroller from the Pi will be in the range of 0mA-20mA at 2.7V-3.9V, which is within the expected range of current/voltage values for inputs to the GPIO pins for Vcc = 2.7V-5.5V.

Equipment: Measurements relevant to the control subsystem will be done using a multimeter. A user-controlled, RC version of the control subsystem will be used to perform unit tests. This RC version will be hosted on the Raspberry Pi. The microcontroller software will be tested and measured using manual control inputs from the Pi. During testing, we will have the Pi generate an extra output at a reserved GPIO pin whenever it interrupts the microcontroller, which will power an LED when activated.

Requirements	**Verification**
The microcontroller will use 6 total outputs to control all of the H-bridges/motors. It will send 2 signals to each of the 3 H-bridges that will enable the wheels to move forward, move backwards, brake, and coast. These inputs will go into the AIN1, AIN2, BIN1, and BIN2 pins on the H-Bridges. Each of these pins require 0-0.7V and 1-30uA in order to operate. The microcontroller will have enough power (from the power subsystem) to send signals within each of these ranges even after the battery can no longer supply sufficient power to move the motors.	For each H-Bridge, run through all 4 commands and make sure the motors/wheels respond accordingly. Additionally, set all 6 control signals to high. If all current measurements and all voltage measurements are within the required ranges for each connection, then the requirements for the microcontroller at high power output are all met.

In order to keep up with the minimum frame rate of the camera, the image processing, state maintenance, communication between the Pi (fetching subsystem) and the microcontroller, the logic on the microcontroller for deciding how to move the motors, and the code to communicate with and use the H-bridges (control subsystem) must take less than 500 ms to execute in order and to completion on a given a single image. We expect the fetching subsystem software to take at least 100 ms on average to execute to completion, and we expect the control subsystem software to take less than 10% of the time taken for the Pi software to execute to completion on average.	Since the microcontroller software's output can't be recorded or displayed on a screen, we will use a special approach for testing it. We will use a camera with a high frame rate (30-60FPS) to record video of ourselves manually sending signals to the microcontroller from the Pi. The moment that the LED lights up will be the start of the microcontroller software's runtime, and the moment that the motors start to move will represent the end. If the approximated difference between these 2 times is less within the specified range of less than 10% of the Pi software's runtime and the sum of the microcontroller's runtime and the Pi's runtime is less than 500ms, then the requirements for the microcontroller are satisfied.

2.3.3 - Sensor Subsystem

This block will use different sensors in order to identify key objects throughout the duration of the fetcher car's procedure. A camera will be used in order to identify the ball, the user's position, and the position of the designated "waiting" point. The ball will be given a special color that is unique to the given scenery so that the camera can easily distinguish it from the rest of the environment and focus in on it. The user and the "waiting point" will also be given unique colors so that the camera can locate which of the 2 it has to travel to after the fetcher car either catches the ball or drops it off. Once the fetcher car gets close to the ball, ultrasonic sensors will be used to figure out when the pincers should close in on the ball.

So, as mentioned above, this subsystem will have 2 types of sensors: cameras and ultrasonic sensors. The camera will be used as the input to the fetching subsystem. It is important to note that we will need to take the frame rate and resolution into account when considering the processing time of the fetching subsystem. In the case of the user and the "waiting point", we can keep the resolution and frame rate relatively low as both of these targets will have little to no movement. The ball, on the other hand, will be moving at varying speeds depending on how it is thrown.

Since we are fetching balls that are often rebounding off of different surfaces, we can guarantee that the ball will not be moving at high speeds. As such, we estimate that the minimum frame rate we can use is 2 frames per second. This should be sufficient in many cases as we are mainly using our fetcher car in indoor environments where walls will keep the ball relatively close to the area of training. In terms of resolution, we have to consider the storage space of the raspberry pi. Since the raspberry pi we are using has 8GB ram, we can ensure resolutions up to 2 megapixels, which should be sufficient for our given testing environment. The system will start to use the ultrasonic sensors when the car gets close enough to the ball, in other words, when

the ball appears to be big enough in the camera, to get an accurate measurement of when to catch the ball. The ultrasonic sensors will take their TRIG inputs from the microcontroller and send the corresponding ECHO outputs back to the microcontroller.

Equipment: Measurements of the effectiveness for cameras and ultrasonic sensors to detect the ball object. We do the test using real simulation, connect the camera module to the Raspberry Pi, connect the ultrasonic sensors to the microcontroller, put the tennis ball in a real basketball court, and test its effectiveness. The test version will be close to the final setup.

Requirements	**Verification**
Since we are tracking moving objects, it is important that the frame rate we choose for our camera is high enough so that the target object's positional data can be processed by the fetching subsystem and sent to the control subsystem with enough time to move the car and keep the object in the camera view. When this is not possible, the framerate should still be high enough that the fetching subsystem can guess the direction that the ball might be in when the ball travels off of the camera.	We will put the ball in several different places on a basketball court, such as right under the basket, near each of the four corners, at the top of the key point, wing, and high post, to simulate the movement of the ball from throwing out of hand, dropping down to the ground, getting picked up by the RC Car, and getting returned to the user.
To make the car able to see both in the front and back, we will have two cameras in total, and a camera multiplexer will be used for connection. In addition to this, we need to have the proper resolution scaling so that we can also differentiate between the pixels of the target we are tracking and the background that the target is located in. Because we have different uses for the front and rear cameras, the front camera will use Logitech C615 which has 1920 * 1080 resolution and 78 degree Field of View (FoV); the rear camera will be Logitech C925e, which has the same 1920 * 1080 resolution and 78 degree Fov, wider enough to detect everything in the back, and acknowledge the car to turn around for a better scan.	For the camera, to lower the risk of system failure, we will connect the camera to the Raspberry Pi with separate USB port connection. First we'll be testing the effectiveness of the rear wide camera's ability to detect the existence of the ball, and we only want a 70% confidence in detecting it. Then, we will connect the front standard camera module, and we want to make sure there is a 95% of detecting confidence.
The ultrasonic sensors need to be able to accurately detect the ball once it is close	We will test the correctness of the ultrasonic sensors up to 40cm. We will first connect the

enough to the car. This is due to the fact that the camera won't be able to fully observe the ball due to its positioning on the car.	ultrasonic sensors to the microcontroller via breadboard or pcb and then test by placing a ball on the ground in front of it (simulating the pick-up process of the car). Starting from 10cm, we will check the accuracy of the sensors and continue to move the ball 5cm away at a time till we reach our 40cm threshold.

2.3.4 - Fetching Subsystem

All of the logic needed in order to utilize the control and sensor subsystems with the objective of acquiring a ball within a given time frame will be implemented in the fetching subsystem. This subsystem will consist of 2 software components. The first will be on the Raspberry Pi. This first component (C1) will be responsible for:

1. Defining the current objective/destination.
2. Identifying the position of the objective/destination relative to the camera view and relaying this information to the microcontroller with 1 byte.

The second software component (C2) will be on the microcontroller. It will be responsible for deciding the direction in which the motors must move in order to get in position to acquire the ball. This will be done by applying logic to the input data from the Pi and the sensors.

Equipment: The measurements taken for the fetching subsystem assess the combined Pi software runtime and GPIO output time. We will do final measurements on the Pi using the attached cameras, but measurements could also be taken using sample images on a machine configured to match the specs of a standard Raspberry Pi 4B.

Requirements	**Verification**
We require the fetching subsystem to operate within at least 100ms in order to give the image processing part of the code enough time to run.	Since the cameras are attached directly to the Pi, the speed of the code can be tested by running the code with a special feature that records and outputs the amount of time taken to fully process an image and send control signals from the GPIO pins. If the code runs for at least 100ms on average and leaves enough time for the image processing and microcontroller software to run to completion within less than 500 ms, then the requirements for the Pi software are satisfied.

2.4 - Software Design

The software design described below all relates to the fetching subsystem as it will be where we process all of the signals we need in order to have our fetcher car function properly.

As described in the fetching subsystem (section 2.3.4), there will be a total of 2 software components that we will consider. The first will be on the Raspberry Pi. This first component (C1) will be responsible for:

3. Defining the current objective/destination.
4. Identifying the position of the objective/destination relative to the camera view and relaying this information to the microcontroller with 1 byte.

The second software component (C2) will be on the microcontroller. It will be responsible for deciding the direction in which the motors must move in order to get in position to acquire the ball. This will be done by applying logic to the input data from the Pi and the sensors.

C1.1 will be implemented as a Finite State Machine. The state of the FSM and the information necessary to handle state changes will be stored as global variables. These variables will be updated whenever an image has been processed completely, and checked whenever image processing starts on a new image. In order to accomplish C1.2 and C1.3, we will divide the images from the camera into upper and lower halves and 3 vertical portions of equal size (total 6 segments of the image). If the objective/destination is within the camera view, then the destination is designated as present in the segments of the image in which it was detected. These designations will be kept in an array of 6 values, which will be used to keep track of the length of time in which the objective was in a particular region, and to identify the regions in which the objective is currently present. In order to keep up with the minimum frame rate of the camera, the image processing, state maintenance, communication between the Pi and the microcontroller (fetching subsystem), the logic on the microcontroller for deciding how to move the motors, and the code to communicate with and use the H-bridges (control subsystem) must take less than 500 ms to execute in order and to completion on a given a single image. We expect the Pi software to take at least 100 ms on average to execute to completion, and we expect the microcontroller software to take 10% of the time taken for the Pi software to execute to completion on average.

States and transitions

State 1 - Wait: It will be assumed upon activation of the design that the car has been placed in its designated location where it will wait for a fetchable ball to enter the camera's view. The car will only transition out of this state when an apparently acquirable ball is detected. The car will return to WAIT once when it has completed a ball fetching task successfully, failed to acquire a fetchable ball within 45 seconds of said ball becoming acquirable (within range of the sensors), or failed to acquire an apparently acquirable ball. For our purposes, a "fetchable" ball is any ball detected at or below a user-selected lower threshold on the camera's view. An "apparently acquirable" ball will be any ball that stays within at least one of the 6 regions on the screen for at least 5 seconds, but can't be reached by the car after 1 minute in the CHASE state. An "acquirable" ball is any ball that can be reached by the car within 1 minute in the CHASE state.

State 2 - Chase: When a ball becomes apparently acquirable in state 1, the design will transition to this state. The car will use the camera to navigate to the ball from a distance outside of the

ultrasonic sensors' range. Once the ball is within a certain (non user-specified) region of the camera's view, the FSM will transition to state 3. If the car can't reach the ball within 1 minute in this state, then the fetching subsystem will transition to the "Return" state.
State 3 - Acquire: When the ball is within range of the ultrasonic sensors and in the non user-specified region of the camera's view (a.k.a when the ball has been reached in state 2), the fetching subsystem transitions to this state. In this state, the car will adjust itself so that the ball is centered in the lower region of the 2 regions at the horizontal center of the screen, and so that the ball's distance from the ultrasonic sensors is small enough that the pincers can close around it. We expect the range of acceptable distances from the sensors to range from 3cm-8cm depending on the final placement of the sensors in the design and the length of the pincers. The car must not spend more than 45s in this state. If the time expires before the ball is acquired, then the fetching subsystem will enter the "Return" state. Otherwise, the car will enter state 4.
State 4 - Fetch: When the ball is acquired in state 3, the Pi will switch to a second, rear-mounted camera to search for the APRIL tag that identifies the user. When the APRIL tag has been located, the car will travel to the user, release the pincers and enter the "Return" state.
Return: When the car fails to complete states 2 or 3, or when it successfully fetches the ball for the user, the fetching subsystem will enter the Return state. In this state, the car will use the rear camera to search for the APRIL tag that identifies the initial waiting location decided by the user. Upon identifying the waiting location, the car will traverse to it and align itself so that the APRIL tag is aligned near the center of the horizontal central region of the rear camera's view.

2.5 - Tolerance Analysis
One of the main problems we might come across is the total weight of our fetcher car. This is due to the fact that an increase in weight will mean that the motors will need to pull more current from the batteries in order to move. If there is too much weight, then the wheels won't be able to rotate quick enough to both move the chassis and push the ball. Additionally, if the current that the wheels are taking is too high, then the battery will be depleted quickly and the total run time of the fetcher car will be very low. As such, we analyzed the current a single motor would require when under different loads. The table from figure 9 shows the current that a single wheel needs when there is no weight on the car.

Voltage (V)	Current (A)
2.5	0.15
3	0.15
3.5	0.15
4	0.15
4.5	0.16
5	0.17
5.5	0.17

6	0.17
8	0.18

Figure 9: Table of Voltage vs Current of the motor (motor rotated freely in the air)

A few things can be noted from the table above. First of all, we can see that the motor began to operate when given 2.5V of power. At this voltage, the motors required 0.15A of current and were rotating at a slow rpm. As we increased the voltage of the power source, the rpm of the motors increased as well, however the current required stayed relatively the same.

When weight was finally added to the chassis, the motor began to pull more current from the power supply. Figure 10 displays a graph of current vs weight at different voltage levels. This data was collected by having a motor connected to the chassis and having the wheel on that motor drive into a wall. We then took note of the current required by the motor as more weight was added onto the chassis. In this stationary position, we concluded that if the wheel was still rotating that it would still be able to push a ball. Now, since the motor was already pushing against a stationary object, the overall current required increased to around 0.21 amps (pushing into the wall without any weight). The required current in this situation differed from when we had the wheels in the air, displaying how making the car push anything already increased the current. With the car in its stationary position, we began to add weight to the car and, as the weight increased, we noticed that the required current also increased.

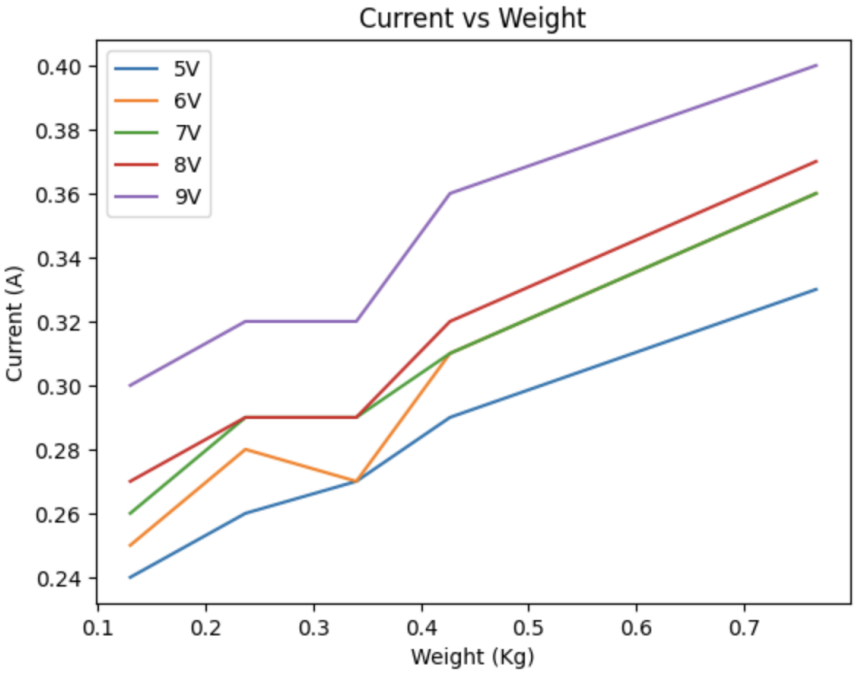

Figure 10: Graph of Current vs Weight at different Voltage Levels

As weight increased, the required current increased as well. Additionally, as the voltage applied was increased, the motors used more current in order to maintain their speeds. Since we plan to

test our fetcher car with balls up to the size of a volleyball, we can conclude that the 4 motors we have will be strong enough. This is due to the fact that a volleyball weighs around 0.28 kg, which is far less than the weight that was tested with a single motor. We also will be able to handle all of the weight added to our car which shouldn't be more than 1kg. Again, we can say this with confidence as a single motor was able to handle weights up to 0.8 kgs with a reasonable amount of current drawn from the power source.

3 - Cost and Schedule

3.1 - Cost Analysis

We found that the typical post-graduate salary of an ECE student at UIUC is about $90,000. Based on this value, the hourly salary we would receive would be about $44. Each individual on our team would make $44/hour x 2.5 x 100 (hours to complete) = $11,000. The total labor cost would then be 3 x $11,000 = $33,000 for all 3 group members.

For the parts we plan on using, the total cost comes out to $352.62 which was calculated using figure 11. Assuming we add additional costs from 5% shipping tax and 10% sales tax, we get a new total of $405.51. Adding up labor costs and part costs, our final project cost would be $33,405.51.

Description	Manufacturer	Part #	Quantity	Cost
Ultrasonic Sensor	SparkFun	SEN-15569	2	$9.00
Logitech C615 Webcam	Logitech	MFR #960-000733	1	$30.00
Logitech C925e Webcam	Logitech	MFR #960-001075	1	$80.00
Raspberry Pi 4 Model B	Raspberry Pi	DEV-16811	1	$75.00
Tenergy Li-ion Rechargeable Battery Pack	Tenergy Power	P/N 31827-05	1	$140.00
DC/DC Buck Converter	TRACO	495-TDN5-0911 WISM	2	$55.00
5k ohm Resistors	TE Connectivity	279-3503G2B5K 11FTDF	3	$4.92

10uf Capacitors	Samsung	187-CL32Y106KCVZNWE	3	$3.27
0.1uf Capacitors	Kemet	80-C1210C106J8NAUTO	3	$3.36
2.2uf Capacitors	Taiyo Yuden	963-HMR316BC7225KL-T	3	$2.07

Figure 11: Table of Components used in Project

3.2 - Schedule

The link below is a Gantt chart that displays the weekly schedule we plan to follow as well as what each individual will be working on. A few screenshots of the chart are displayed in figures 12 and 13 to show off a quick view of what they look like.

Link:
https://docs.google.com/spreadsheets/d/11F2bXZBZZVbTI17Cs0ifYs-3oXvgZa6a/edit?usp=sharing&ouid=103313886692512844099&rtpof=true&sd=true

Images:

Initial setup				
Establish details: Design each subsystem, figure out car layout, figure out motor positioning	Everyone	100%	3/24/23	2/12/23
Acquire hardware: Get car chasis, Pi, motors	Everyone	100%	2/12/23	2/15/23
PCB design: Figure out initial board components	Everyone	25%	2/15/23	2/19/23
PCB design: Produce a near-final design of board	Everyone	50%	2/19/23	2/24/23
PCB design: Create the pcb layout and connect all of the components	Everyone	100%	2/21/23	2/25/23

Figure 12: Screenshot of Gantt Chart

Breadboard prototype				
Power: Create circuitry for power system, calculate required voltages and currents, test motors with different voltage/current levels	Patrick, Luis	25%	2/25/23	2/28/23
Control/Sensor: Implement code for control subsystem (code for the microcontroller to function), implement code to allow ultrasonic sensors to function (test as well)	Luis, Patrick	50%	2/28/23	3/4/23
Fetching: Implement code for image processing, implement code for FSM and handshaking with microcontroller	Luis, Yinshuo	100%	3/4/23	3/12/23
PCB redesign: After code testing and breadboard testing, adjust components on pcb design	Everyone	75%	3/10/23	3/13/23
PCB redesign: Remake pcb layout and reconnect all components (if needed)	Patrick	100%	3/13/23	3/16/23

Figure 13: Screenshot of Gantt Chart

4 - Discussions of Safety and Ethics

Our experiences with working on a design project of this kind are limited and, as such, we will have to make sure that the work we undertake is assigned appropriately to each individual in the group according to their experiences and/or their desire to learn. However, this will make it the responsibility of each individual in the group to clearly state the limitations of their experience, knowledge, or other factors that would contribute to completion of the project before the due date. This falls under IEEE code of ethics section I.6 that states that we should "maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations." With our project, there are concerns relating to our pincers. Since they will be powered by motors that have a decent output torque, we have to make sure that they don't injure any users or potential bystanders. We will do so by designing rounded pincers that won't be able to cut or pierce any individual. On top of that, we plan on closing the pincers at a slow enough pace so that individuals near them have enough time to back away (i.e., pull away their fingers if they see the pincers closing). Following such procedures will ensure that we follow IEEE code of ethics section I.1 that states that we should "hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design."

In addition to the IEEE code of ethics, we plan on following sections 1.1 and 1.2 of the ACM code of ethics. That is, with our project, our only intentions are to contribute to society and not produce a negative output with our results. In addition, we plan to avoid as much harm as possible. This includes both physical/mental injury to the individual operating the car and environmental damage where the fetching car is operating.

There are a few safety concerns that we have to consider when it comes to all of the components of our car. As mentioned in the first paragraph, we are dealing with motors that have the potential for high torque movement. As such, to reduce any physical harm to anyone using the car, we will make sure that the pincers are round and move slow enough such that no harm can be done by them.
In addition to this, we also have to consider the harm that can be caused by high currents running throughout our circuit. Luckily, such high currents are only limited to a single part of our design: the wheel motors. The circuitry for these motors is limited to only one side of the pcb. As such, to protect any users from harm, we plan on holding the pcb in a case so that users won't have to touch it directly. Additionally, we will make sure that there are no open wires that users could touch to harm themselves while the car is operating. Since the connections to the motors are below the car, we can guarantee that no random obstacle or individual will be able to get near the wires and entangle with them.
Finally, we have our battery to consider. It has the potential for a maximum output voltage of 11.1V as well as the potential to emit 5.6A. Due to this, and since we are using a Li-ion battery, the battery can get very hot and can harm individuals with both current and heat. To ensure this doesn't happen, we plan on hiding the battery in the middle section of our car where no user will be able to touch it (since it will have both a plastic panel above it and below it). In addition to this, the battery will not be supplying more than 6V at any given time as that is how all of our circuitry was designed. As for the current, we will ensure that the battery is connected securely

to the pcb so that no one can touch any open wires. Since this is the only place where it will be connected to, we don't have to worry about too many wires protruding from the battery.

Since our car will be moving throughout a populated environment, it is vital that we minimize the risks that the car can cause to any individual and to the environment itself. With all of the precautions we will be taking, both with the physical design of the car and the circuit design of the car, we are confident that the car will be safe to use and safe to be around.

References

Institute of Electrical and Electronics Engineers. "IEEE Code of Ethics." Available: https://www.ieee.org/about/corporate/governance/p7-8.htm [Accessed: 20-Feb-2023]

Association for Computing Machinery, "The code affirms an obligation of computing professionals to use their skills for the benefit of society.," Code of Ethics, 2018. [Online]. Available: https://www.acm.org/code-of-ethics [Accessed: 24-March-2023]