Bumper Car Sumo

Jaden Simon - simonjaden223@gmail.com
Melvin Bosnjak - meco0597@gmail.com
Daniel Humeniuk - humeniukdaniel@gmail.com
http://bumpercarsumo.weebly.com/

***Abstract*— BumperCarSumo is a fully interactive multiplayer game. Each player controls a little robot on a designated play area. A robot is controlled by a central game hub that communicates to the robots via WiFi. The controllers are connected to the hub with the use of Bluetooth technology. The robots are powered internally with a rechargable lithium ion battery and moves about the play area with geared DC Motors. The goal of the game is to be the last remaining player on the play area by any means possible. Once a player has fallen off of the play area, the player has lost the match. The hub tracks each robot with a web camera and an OpenCV program to ensure that once it has fallen off the play area, the robot will turn off and remains off until the next game. Once all but one player remains, the round is over and is free to be played again.**

I. Introduction

The purpose of BumperCarSumo is to provide entertainment that mirrors popular Battle Royale game play with the use of physically controlled objects. The components currently required for game play is a white play area, a dark (black if possible) out of bounds zone, a web camera, a Raspberry Pi 3 B+, the BumperCarSumo program loaded onto the Raspberry Pi, two to four remote controlled robots, and the corresponding amount of controllers. Each component will be described throughout this report.

II. Background

The concept of BumperCarSumo originated from a desire to control robots and stage a free-for-all match between players. Battle Royale video games have become a huge success in recent years and we wanted to attempt similar style of ”last man standing” with these remote controlled robots.

The concept of the robots came from the Ollie Sphero. The comparison of the Ollie and our design can be seen in Figure 1. Our design came out much bigger than expected due to the gears required to slow the motor down. The motors also created another reason for more space. The material used for traction was much simpler as we were intending for the robots to be able to slide if hit but still able to maintain a grip on the play area.

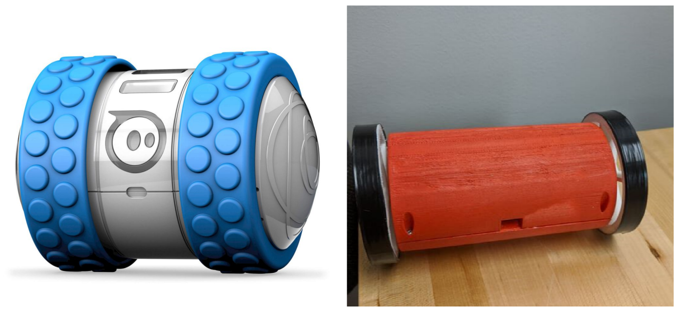

Fig. 1: The Ollie from Sphero comparied with our Design
Source: Adapted from [1]

III. Project Implementation

The central piece of the game is the Game Hub. This Hub is constructed with a Raspberry Pi 3 B+ which included enough features to make the gameplay possible. The robots are PCBs and DC Motors wrapped in a 3D printed shell. Each PCB receives commands from the Game Hub via WiFi. Players control each robot with the use of a Wii Balance Board which is connected to the Hub through Bluetooth. The web camera is connected to the Game Hub with USB and detects the unique color of each robot. The play area is a 4ft round surface in which the robots are able to move around and play. The camera sits above the play area on an 8ft post. Each component of the project is described in more detail below.

A. The Game Hub

We utilized a Raspberry Pi 3 B+ as our game hub. The device contained both Bluetooth and WiFi capabilities which was critical for communicating to the external game peripherals. The Pi was an excellent choice as it is small, portable, and capable enough to handle the software required for the game.

The controllers were connected to the Game Hub via Bluetooth. The controllers are based on Wii Balance Boards. The Game Hub starts the controller logic in a separate thread and once connected, continuously polls the Bluetooth devices to ensure they stay connected. When a game is in progress, the data collected from Bluetooth is processed. Once processed, the Game Hub will determine in which direction the player is leaning and send the corresponding command to the player's robot.

B. Controllers

For our controllers, we used the Wii Balance Board. Our intention behind this controller system was to add layers of complexity and extra challenges to gameplay. The interface was an open-source library found on github. There were a few versions of the library but we electected to use the library written in Python. We made an API for our purposes to be as simple as possible with only two functions that connected to a board and collected the data from the board.

Our API relied on a single WiiController object that connected to multiple boards. The functions of the API are start and get_data. The start function accepts the MAC Address of one of our boards and instantiated the board to a corresponding WiiBoard object board1, board2, board3, or board4. Once connected, the board will need to be continously polled to ensure the Bluetooth stayed connected. The get_data function accepts the MAC Address of one Balance board and an array of that board's data would be returned to the Hub. The Hub then processes this data and sends the proper commands to the corresponding robot.

C. Robots

The robots are a mix of hardware and software. A PCB lies within each robot that contains an ESP-12F. The ESP-12F runs custom software that drives the robot. We printed custom shells and wheels to protect and drive the robots. The custom hardware and software made debugging the robots a challenge but a worthwhile effort nonetheless.

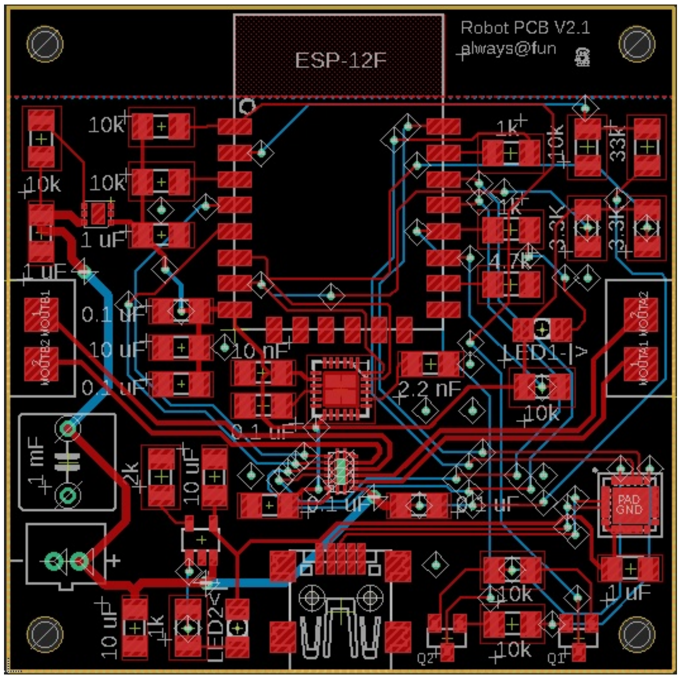

Fig. 2: The custom PCB that controls each robot

The custom hardware lies in the PCB which can be seen in Figure 2. The PCB contains 4 mounting pads, 2 for each DC Motor that drives the robot, as well as M2.5 drill holes on each corner. The drill holes allowed for the PCBs to be mounted within the shell as seen in Figure 3. The ESP-12F chip on the PCB was programmed with the use of USB and a CP2104 programmer chip on the board. An MPU is located in the center of the PCB. The MPU was integrated as a backup to the camera detection. The data from the MPU would allow us to determine if the robot had fallen from the board. This was later discarded as our Computer Vision portion of the project worked as expected. A DRV8835 motor driver was used to control the motors, capable of delivering up to 1.5A per motor. A voltage regulator was used to convert the higher battery voltage to a steady 3.3V.

The software in the ESP-12F is responsible for receiving data from the Game Hub via WiFi to drive the motors. Some additional functionality was also present, such as monitoring battery levels. Static IPs were set up for each PCB to ensure that they remained consistent whenever the game was setup. A better way of doing this would be some sort of discovery protocol. The integration of both hardware and software on a custom PCB was a challenging and rewarding endeavor.

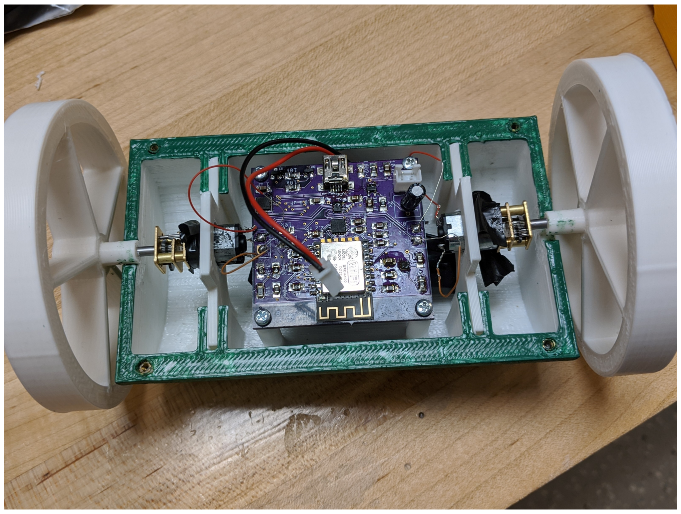

Fig. 3: The finalized robot components in shell

D. The Camera and OpenCV

The camera used for the project was a Logitech web camera. We considered the Pi Camera as well for the camera but proved to be unreliable. Our first experiment with the Pi Camera ended up shorting something out in the circuitry and our first Raspberry Pi was inoperable. The Pi Camera did offer faster operating speed as the USB interface was bypassed but we elected to use the Logitech camera as it was safer. We used a 12ft male-to-female USB extension cable which allowed for the camera to rest on top of the camera perch and not the entire Raspberry Pi system which allowed us to easily use the Pi on a nearby table. After a few tests conducted with the camera and OpenCV, any fears of latency were soon removed as the camera proved to be an easy to use asset perfect for the job.

The OpenCV program that we used and tweaked captured color only on a white mask and held each color to a pixel threshold to determine if the robot was in or out of bounds. When the game began, the play area needed to be empty. The camera would sample the play area a few times and create a black and white mask. The white play area would be captured and all other colors would receive a black mask. Everything within the black mask would be undetectable regardless of color. Each robot color had an associated color hue with a range that would allow each robot to be detected. Each color had a minimum pixel count to be considered detected. Once a player falls off the board, the pixel range for that particular robot would be under this threshold and the Game Hub will send a stop command to the robot and no longer treat the associated controller's input as valid.

E. The Play Area

The play area is simple in nature but plays an essential part in the overall game. We painted the table top white. The floor we were working on captured the carpet as black, avoiding a need to place a black sheet as the backdrop. As described in the camera section, every color that would enter the black region would be filtered out, meaning once the robot was off the white play area, it turned off and was unable to accept user input. The most complex piece of the play area was the 8ft pole which held the camera. Figure 4 shows how high up the camera needed to rest in order for the entire play area to be seen. It was essential the camera be at this height as to ensure all robots in the area would be captured. We were lucky enough to get a table that sat off the ground enough so that once a player was out, they could not reenter if a glitch with the Computer Vision were to occur. Though this proved to be a nonissue, it was reassuring to have a backup in the worst case scenario.

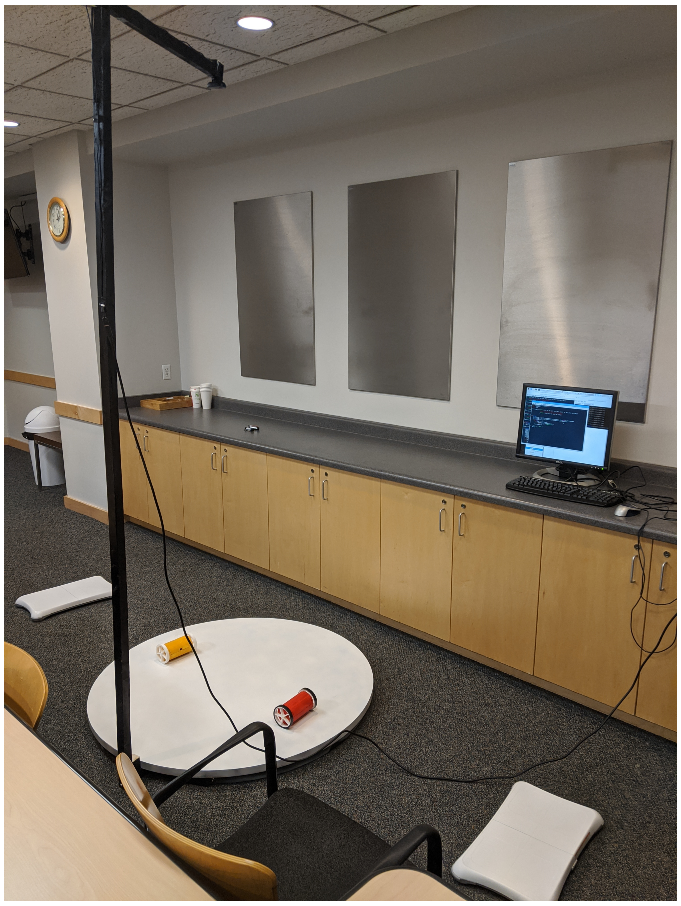

Fig. 4: The constructed play area

IV. Discoveries

We learned pretty quick that the original DC motors we were using were far too powerful for such a small play area. The motors offered over 1000 RPM which can be handled with gearing the motors. The unfortunate result of this, however, was the small area in which the gears were to fit. The first few attempts at gearing the motors were a success in many ways and a failure in others. The gears did work and we were able to get a simple gear test box working as seen in Figure 5. The gears faced problems with proper meshing, resulting in small pieces of PLA to fly from the test box. With lubrication, the problem continued still. We also attempted acrylic gears as a laser cutter was able to make more precise cuts, allowing the gears to mesh better. Similar problems persisted with acrylic gears as well and any lubrication on the acrylic gears proved to be fruitless.

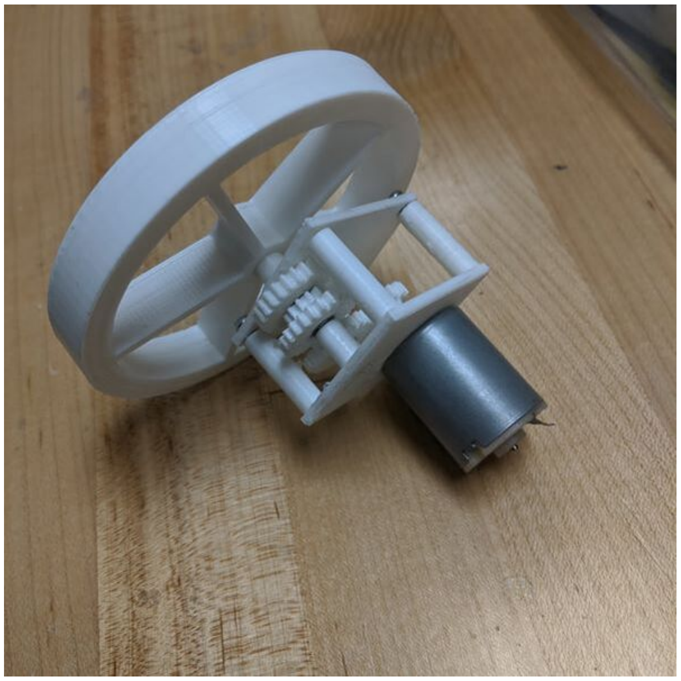
Fig. 5: The first ideration of the designed gears

The biggest shortcoming we encountered was not developing a prototype in early November. We ordered some of the pregeared motors in October and elected to go with the more powerful motors that required custom gearing to work. In retrospect, we saw that even though we did not intend to go with the pregeared motors, we could have and should have created a prototype with the pregeared motors to test the game area, controllers, and OpenCV with a working robot. This would have also saved time after determining that the custom motors would not work as expected.

V. Evaluation

Despite any setbacks and initial concerns, the project was a huge success. While we were not able to accomplish all of our stretch goals, the most basic of play modes was achieved and everyone at the Demo Day had fun playing the game. There were many areas of the project that had unique hang-ups, challenges, and roadblocks. Some of these we were able to solve with little to no issue, others took some time, and the leftovers were abandoned. The next section will cover aspects of the project that worked well and others that are now dead in the water.

The gears are among the greatest achievements and failures of the project. If they had succeeded, the robots would have had more power, leading to harder collisions and potentially more intense games. After a few tests with a completed robot with the custom gears, we soon discovered that the gears proved to be unreliable. The lubrication did stop the gears from heating up and melting but the inconsistent gear meshing led to an unsuccessful implementation. While testing the custom geared robots, the gears would not always turn and got stuck easily. This would have been a disaster during Demo Day. After a few days of trying to solve the issue and attempting to make acrylic gears, we decided to ditch the gears and use pregeared DC Motors. The pregeared motors did not offer the same power but they were able to get the robots moving consistently. The gears were devastating to the project and is the main reason why our stretch goals were not met.

The gears not only cost us extra time, it cost us extra money. The original motors purchased totaled at $12.00. With it, brass rods were required along with lubricant. Once acrylic gears were tried, three acrylic sheets were purchased, totaling around $16 as well as adhesives. The gears required several iterations of shell components which resulted to a loss of our PLA filament. With the excess filament, we could have increased the amount of players on the board. The original motor and gear scheme took up a lot more room than the later design, leading to larger shells which required more filament. The gears did end up costing us more but fortunately the expenses were not too excessive. We are not mechanical engineers.

Despite the gears being a loss, the design of the shell proved to be a success. The design was modular in nature. As can be seen in Figure 6, there are inserts for the gear plates that were later modified to suit the needs of the pregeared motors that were used in the final design. The batteries used in the shells weighted it down enough to prevent the body of the shell from excess rotation. Ramps were also added to fit in where the square hole was that was originally designed for USB cable insertion. This was a fortunate coincidence for us as the ramps prevented the robots from rotating the body forward and also added an extra game mechanic, allowing to move other robots from the side and not the only the front. Though we found that the body will still rotate with the ramps when three or more players were jumbled up, it has proved to be a fun ”feature, not a bug” game mechanic. The wheels required a small piece of electric or duct tape for good traction on the play area. Fortunately, the cost of the shell was limited to the cost of the PLA filament which totaled no more than $40. The shell was designed well and the modular design saved us as many changes were made later in timeline.

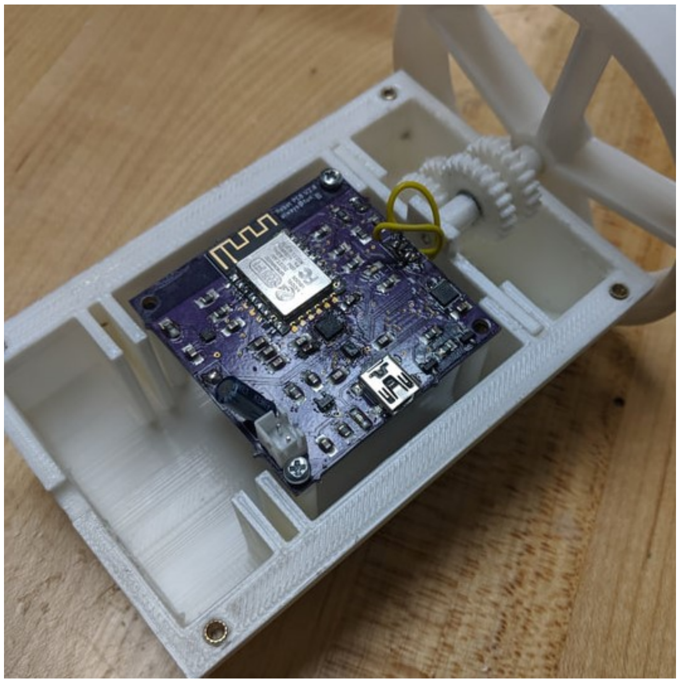
Fig. 6: A snapshot of the insides of the original robot shell with gears

The PCBs used for the project were well designed with each iteration and worked well every step of the way. The first PCB was designed in April, right before our planning phase was complete. This preliminary PCB was equipped with pinheaders (absent in later models) to debug the design. This worked in our favor as we discovered that the MPU held some pins low by default which prevented the ESP-12F from powering up. We were able to make this discovery by utilizing I2C to communicate with the chip and turn the pin to high to activate the ESP-12F. The second PCB corrected this by removing the connection of that pin to the ESP-12F. This was a trade off as we were no longer able to receive interrupts from the MPU. The second PCB also contained drill holes for mounting the PCBs into the robot shells as well as an optimized placement of components. The DRV was moved to the middle of the board with the MPU for the purpose a more even driving power. The third and final PCBs were very similar to the second with a few parts moved around to ensure each component was clear of the screws in the drill holes. The PCB, while a tedious task to solder, was one of the most successful components of the project that required very little change.

The Raspberry Pi was a very useful aspect to the project, allowing us to use free and open source libraries, but we were ill prepared for the issues it came with. The Raspberry Pi worked very well for our needs and the multicore processing power helped the game logic run smoothly. Melvin was able to enable threading so the Bluetooth controllers, WiFi, game logic, and camera/OpenCV interface ran seamlessly. We were unaware of the poor power protection provided by the Raspberry Pi hardware. The first Raspberry Pi was fried after the Pi Camera drew too much current which also broke the Pi Camera. We assumed incorrectly that the Raspberry Pi would have a robust power protection scheme. The second Raspberry Pi was fried for unknown reasons. The third Raspberry Pi fried when two GPIO pins shorted to one another. Once we discovered how vulnerable the Raspberry Pi’s are, we treated the last with extra special care. The final Raspberry Pi was to remain in the casing and only be connected to anything via USB, HDMI, and the official Raspberry Pi power supply. Once these issues were all sorted out, we treated it as delicately as a newborn baby. We were on the verge of porting all of the code to someone’s laptop but the laptop would require running Linux as all of our Raspberry Pi code works only on Linux libraries. This would have been an even larger upset so we decided to keep with the Raspberry Pi and we fortunately found out all potential power issues and it suited our needs.

The Wii Balance Boards made the game play a lot more fun than intended. When selecting our controllers in our initial proposal, we stated concerns about multiple Bluetooth objects in a single area creating a lot of noise. This proved to be a nonissue as there were little to no issues with interfacing with the Wii Balance Boards. We used a simple control scheme to make the game play intuitive, fun, and exciting. When a player leans forward, the robot moves forward. When the player leans back, the robot moves back. When leaning left and right, the robot would only rotate in that direction. This added some extra strategy from a player standpoint. Players need to now think a few moves ahead to ensure that they will not be vulnerable when needing to move right or left. The controllers proved to be a vital component that made the game so enjoyable.

The use of WiFi and Bluetooth to control the game was a huge success. During our presentation, we noted that were was a lot of potential for noise interference and latency issues. Fortunately we did not encounter any of these issues. The response time from controller input to robot movement was so good there was hardly any latency observed. The response time of the camera and OpenCV was also fast enough that when a robot left the arena, it was immediately turned off.

VI. Bill of Materials

```
Per PCB:
1x DRV8835 low-voltage motor driver
1x JST 2 Pin SMT RA connector
1x MPU 6050 IMU
1x SMT USB Mini-B connector
1x MCP73831/2 LiPo battery charger
1x TLV757P LDO
1x ESP-12F
1x CP2104-F03-GM
7x 10k Ohm
4x 1k Ohm
1x 4.7k Ohm
1x 2k Ohm
3x 3.3k Ohm
1x 10k Ohm 1\%
1x 33k Ohm 1\%
1x 100 uF electrolytic capacitor (through hole)
5x 0.1 uF ceramic capacitor
3x 1 uF ceramic capacitor
3x 10 uF ceramic capacitor
1x 2.2 nF ceramic capacitor
1x 10 nF ceramic capacitor
2x LEDs
2x NPN transistor Package: SOT23

1x Table with no legs
1x Logitech web camera
1x Raspberry Pi 3 B+
2kg PLA
Spray paint
Various screws
```

After totaling each invoice, we have determined that our project has cost approximately $860.

VII. Conclusion

Our senior project proved to be a huge success and we learned many valuable lessons from it. We were able to work the first three months of the semester in parallel which resulted in rapid development. We began integrating around November and had it finalized just in time to present at Demo Day. The project milestones and meeting logs can be found at http://bumpercarsumo.weebly.com/.

References

[1] Ollie. Accessed: March 6th, 2019. [Online]. Available: https://www.sphero.com/ollie