ARio Kart

Authors: Sourav Panda, Ranganath Selagamsetty, David Yang

Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—An augmented reality racing game with networked remote controlled vehicles. The basic concept of the game is similar to Mario Kart, with a slalom-style user-specified race track and various virtual items and power-ups to augment the physical car race. The system consists of cars, gates, and computers that run the game software and render the AR display and interface. Compared to similar existing AR games, this game would provide a more engaging AR experience with meaningful interactions with the environment.**

***Index Terms*—Augmented Reality, Computer Game, Computer Vision, Infrared Detection, Object Detection, Remote Control**

I. Introduction

A new frontier in human-computer interaction, augmented reality is a technology that uses an interactive display of the physical environment to enhance the user's real-world experience [1]. Augmented reality has the potential to transform everyday scenery into an information rich environment for new immersive gameplay experiences. Modern augmented reality games fail to reach the potential of the medium as many of them confine the user experience to static interactions with the background. This often amounts to traditional games simply being overlaid onto a video stream without taking advantage of the visible features of the current environment.

The solution proposed in this paper, ARio Kart, aims to improve upon existing AR games by unifying physical components that interact with the surroundings and game software that augments the display. ARio Kart is a slalom-style racing game with physical cars and gates and virtual items. Users interact with game software on their laptops to operate remote controlled cars and see an augmented video stream from the cars' perspective. Though related products, such as drones and ordinary RC cars, exist, they have many shortcomings, including their cost and battery life, and lack the AR aspect. ARio Kart will provide all the fun of traditional RC cars with the added immersion of the first-person Augmented Reality component.

II. Design Requirements

In order to deliver a high-quality AR racing game experience, the implementation should meet the following software and hardware design requirements.

A. Software Requirements

The primary design requirement for the software component of the project is to ensure a seamless user experience. As such, the video streaming component should deliver at least 720p 30fps video with a latency of no more than 100 ms. This value was derived from the related field of online gaming, where 100 milliseconds is considered the upper bound for acceptable latency in gameplay. A delay greater than this could induce nausea as well as frustration due to lack of responsiveness. This will be tested by measuring round-trip time within the user and the car, and extrapolating one-way latency from this.

An additional requirement is making the game robust to user input, which will be tested with unit tests in the game software. To test the overall quality of the experience, we will play the game.

B. Hardware Requirements

To guarantee that the physical vehicles and gates can provide system software information about the state of the game, the following requirements must be met.

Each car must must have a controller that can set the motor speeds with ± 2 rpm (derived as an acceptable tolerance given the size and the 4.5 mph top speed of the car) of a desired value.

$$error = \frac{2\ rpm * 2.56\pi\ in/rev}{4.5\ mph * 63360\ in/mi\ /\ 60\ min/hr} = 0.34\%$$

This allows standardization in speed control and ensures that no car has an unfair advantage due to hardware discrepancies. This requirement will be verified by comparing the speed measured from a motor encoder with the desired target speed.

Each car will have an Radio Frequency Identification scanner mounted to the bottom of the car that will be able to detect a gate before the car completely passes over it, a latency of no more than 81 ms (derived given the size and the top speed of the car as well as the location of the scanner on the car, 6.4 in before the rear bumper).

$$latency = \frac{6.4\ in\ to\ rear}{4.5\ mph * 63360\ in/mi\ /\ 3600\ s/hr} = 80.8\ ms$$

This allows the game software to accurately determine which place each car is in after passing through a gate.

Each car will have an Infrared Emitter that should be able to detect an Infrared Receiver mounted on another car with variable degrees of accuracy at different distances (must ***at least*** meet the following profile: 99% at 2 m, 95% at 5m, 90% at 7m, 85% at 10m). This will allow cars to detect opponents ahead for using items. This requirement will be verified be placing the receiver circuit at various distances and recording how many times the receiver can successfully detect a pulse from the emitter.

III. System Architecture and Interactions

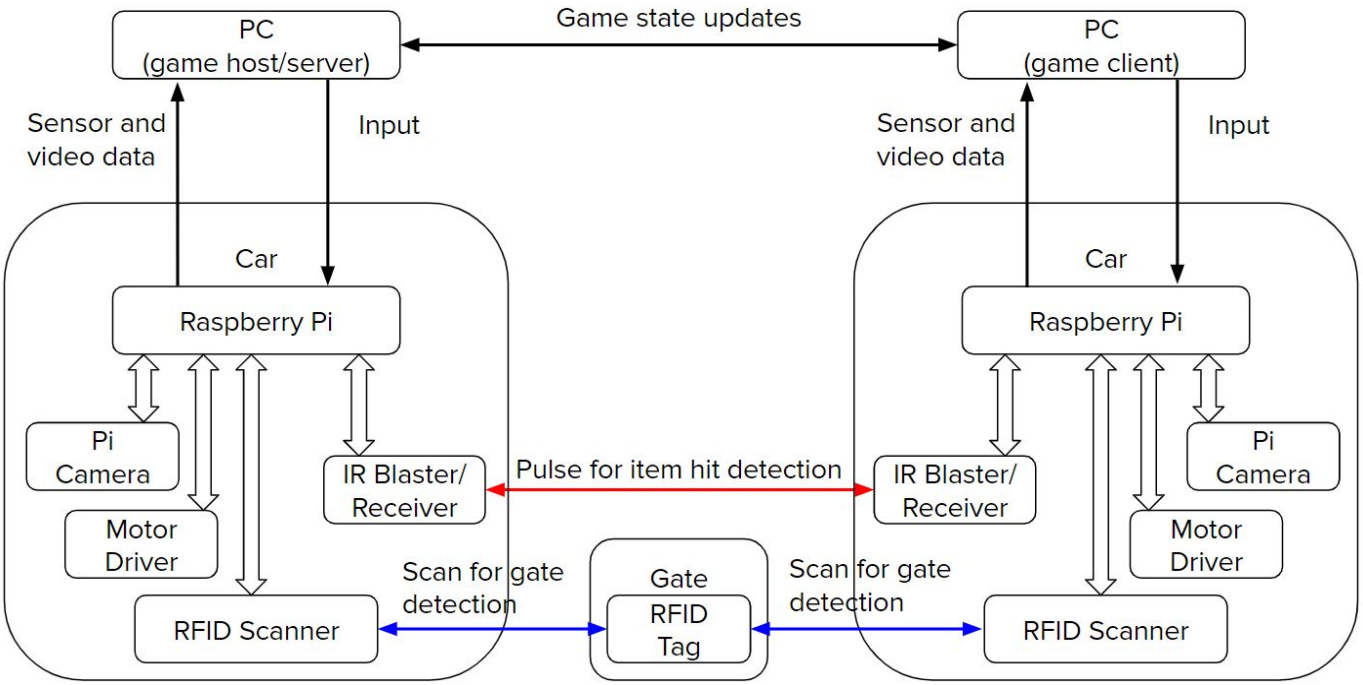


Fig. 1. Overall system architecture block diagram

A. *Overall Architecture*

The general design of the system follows the architecture depicted in Fig. 1. From the highest level to the lowest, the physical components are PCs that run the game software and display the AR overlay, Cars with Raspberry Pis that control the motor, IR, RFID, and camera modules, and gates that are scanned by the cars. Each PC is connected to one car and the game host. There are four major communication pathways between the physical components in the system as shown in Fig. 1, with communications over Wifi in black, IR in red, and RFID in blue:

1. *Between two PCs*: Each PC runs an instance of the game, with one acting as the game's host and the others as clients connected to the host. Communication between PCs consists of game state updates with item use and gate passage information from their respective cars.
2. *Between a PC and a car*: Communication from the PC to the car consists of player input with commands for the physical components on the cars, such as commands to set motor speed or commands to fire the IR blaster. The car sends to the PC a constant stream of video and discrete sensor data from the RFID and IR modules.
3. *Between two cars*: A car following another car in a race can fire its IR blaster at the IR receiver on the car in front, causing it to stop if a valid item was used.
4. *Between a car and a gate*: Cars passing over gates will scan the gates to determine their position in the race and on the course.

Detailed communication diagrams for the two major interactions of item usage and gate scanning follow.

B. *Item Interaction*

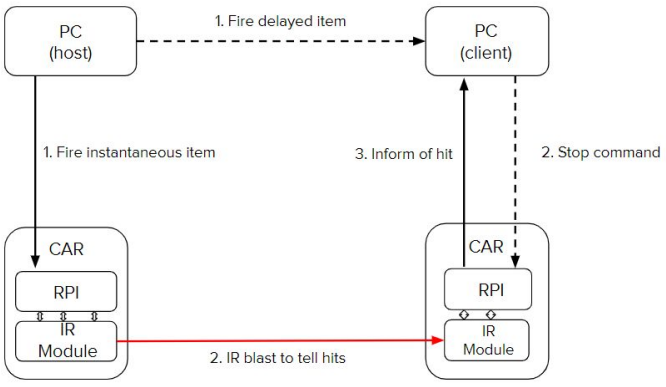


Fig. 2. System interactions for item usage

The flow of interactions depicted above follows two different paths depending generally on the type of item used: instantaneous items and delayed items. Instantaneous items, when fired by the user, take immediate action, sending a

command down to the car to fire a IR blast. This blast, if a hit, will be registered by other Car. A delayed item flows in the opposite direction. Some items require greater knowledge of the game state, for instance place, and therefore must be passed through the PC Host. These are then forwarded to the target. The target PC then forwards the correct consequence to the car.

C. *Gate Interaction*

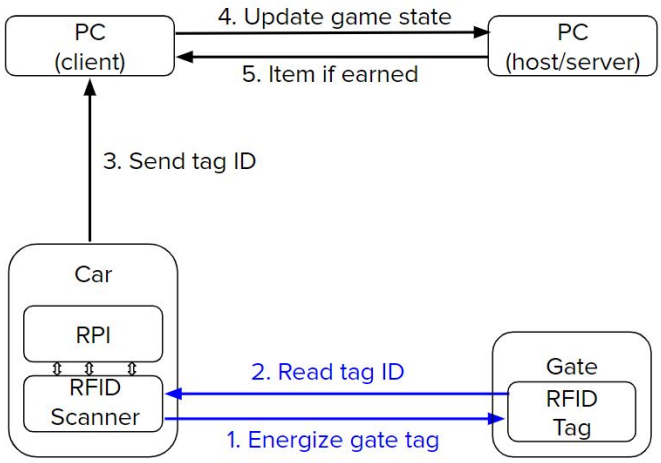


Fig. 3. System interactions for detecting passed gates

When the RFID Scanner in the car first passes over the gate, the RFID Scanner will energize the tag and read the bit pattern of the tag ID in the gate. This gate ID is then wired to the onboard computer. The onboard computer pass this ID to the the PC that it is networked with. The PC relays this information, as well as as a timestamp of when the ID was received to the other PCs running game instances to be able determine what place the car is in. Since game-item distribution happens only when a car passes over a gate, the "host" PCs will provide the "client" PC an item, if that user has earned a item.

IV. Design Trade Studies

A. *Car Detection*

Originally, we had planned for only using an assortment of sensors to detect if an opponent car was directly in the line of fire of a user's car. To be able to do this, we would have needed the sensor array to be able to transmit data related to which car shot the item and what shot the item. This would have allowed our PC servers to have knowledge of which cars were affected by which items. This, in turn, would have allowed our game to have more complex animations for item usage (ie, seeking missiles, area-of-effect items). However, we found that the circuit design for this type of data transmission was far beyond what we had the time and resources to do. For data transmission, we would've faced issued regarding clock synchronization between two independent systems in motion and require packet transfer occur extraordinarily fast. Alternatively, we decided to simplify the interaction to the point where a car simply needs to know if it was hit. We chose to split car detection between an IR sensor array in addition to computer vision on the video feed. IR is the preferred method of transmission and detection as it is cheap, easy to design circuitry around and is not a focus beam. These factors mattered to us as our expenses push us very close to our limit, we did not want to invest significant time in circuit design on a sub-system of the project and so that we did not need to have a large array of receivers on the cars. Since the cone of transmission for an IR LED is very wide, we were allowed the benefit of reducing the number of receiver photodiodes on each car, thus reducing our cost and weight per car. Thus, we believe that a combination of IR sensor and camera dependent items would yield the best user experience.

B. *Steering System*

A trade-off was made between a more accurate Ackermann steering system and a Differential System. For our project we chose to go with a differential steering system, as the Ackermann steering system added mechanical complexity to an already mechanically complex project. Ackermann steering kits were also much more expensive than their differential counterparts. This, however, came at the cost of accuracy in turning.

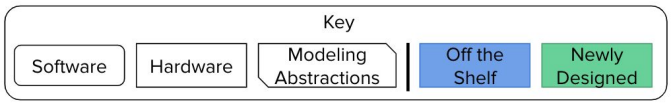
C. *Communication Protocol*

To meet our latency requirements for data transfers between the PCs and Pis, we considered various communication protocols, including TCP, UDP, and RTSP. TCP is something we all had experience with, and its reliability and ubiquity make it easy to work with. However, our research showed that most modern live video streaming applications use some form of UDP-based protocol due to its faster transmission with less error correction overhead. Among the UDP-based alternatives, the Real Time Streaming Protocol (RTSP) looked promising, but we were unable to find high quality libraries that supported it. UDP on its own would require us to write our own error detection and correction, which would be difficult because the video stream is compressed. We chose TCP because of its fast development time and our experience with it, but we could potentially switch to a faster protocol if we are unable to meet our latency requirement with TCP.

D. *Workload balance*

As our project includes Raspberry Pis in every car as well as PCs, naturally there were two places that we could have done our computing. We decided that most of the compute would be done on the more powerful PC's, and the Raspberry Pi would solely act as a slave device. This was done due to the large difference in compute capability between the two devices, however at the cost of higher network traffic and a more complicated server structure.

V. System Description



A. PC Software

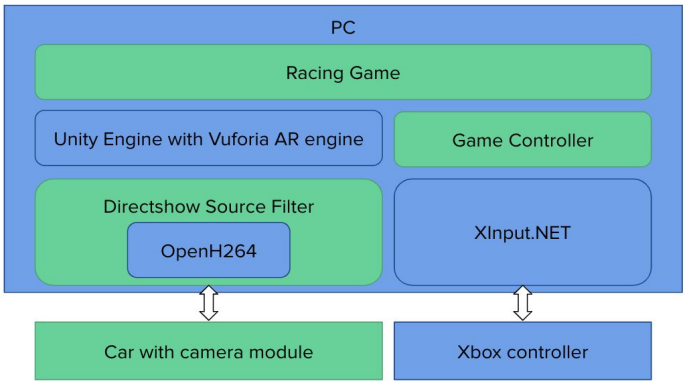


Fig. 4. Subsystem diagram for PC

The software run on the PCs can be separated into three modules: game, video, and controller.

1. *Game*: The game software will manage state and operations related to the race including information about items and player place and progress. It is also responsible for displaying a car's video stream and rendering an AR overlay with race and item information. The game will be implemented in C# using the Unity game engine, with the integrated Vuforia engine for AR support, and Unity Multiplayer for multiplayer matchmaking and communication between game instances.
2. *Video*: The video module is responsible for receiving the video stream from the Raspberry Pi on the car and rendering it to the screen in a way compatible with the AR engine. For this implementation, this module takes the form of a C++ Directshow source filter, which uses the Directshow Windows API to act as a virtual camera device recognizable by Vuforia. The source filter receives raw H.264 compressed video from the Pi through TCP/IP over Wifi, decodes it with the OpenH264 library, then draws it to the screen.
3. *Controller*: The controller software receives input from the player, converts it to commands for the car, and sends those commands through TCP sockets to the car, which then controls its physical components. Players interact with their PCs through Xbox controllers, which interface with the controller software using the XInput.NET C# library. The controller module also receives RFID and IR sensor data from the Pis, which leads to game state updates in the game software.

B. Onboard Software

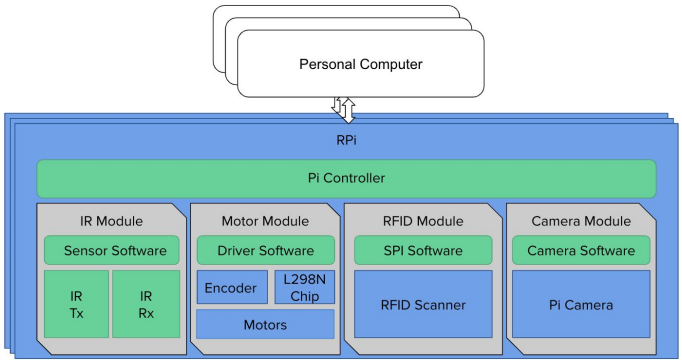


Fig. 5. Subsystem diagram for Onboard Controller

The software run by the onboard computer and hardware components on the cars can be separated into five discrete modules: Controller, IR, Motor, RFID, and Camera.

1. *Controller Module*: Each Raspberry Pi will have software that receives data from the four other modules in the car and transmits this data over Wifi to the PC that the onboard computer is networked with. The controller software will receive inputs to control the movement and actions of the car, whether the IR receiver was triggered, the speed from the motor encoder, the gate ID when passing over a gate and the video stream from the camera. The controller software will transmit whether the IR receiver was triggered, the gate ID when passing over a gate and the video stream from the camera to the PC it is networked.
2. *IR Module*: The Raspberry Pi will have IR sensor software that interfaces with the IR circuits on the car. This software will receive commands passed from the PC through the controller software instructing the car to fire the IR blaster. The IR sensor software will drive voltages to pulse the IR blaster. This software will also alert the controller software if the IR receiver circuit has been triggered by an opponent's IR blast.
3. *Motor Module*: The Raspberry Pi will have software that interfaces with the motor's encoder chip as well as the motor driver chip. It translates commands from the PC into voltage levels for the L298N motor driver circuits. The controller software, motor driver circuits and Hall-effect encoders create a feedback control loop that allows the onboard computer to accurately set speed by modifying the PWM signal given to the motor driver circuit.
4. *RFID Module*: The Raspberry Pi will have software that interfaces with the RFID scanner on the car. The controller software will interact with the circuit via the SPI protocol. This protocol allows for high speed

data transmission and is directly applicable as the system follows a typical master-slave configuration. The RFID scanner will be continuously reading and will only alert the controller software when the scanner has successfully detected a NFC tag ID from a gate.

5. *Camera Module*: The Raspberry Pi will have software that interfaces with a Pi Camera V2. The camera supports live-streaming video in h.264 video encoding format. The module is nothing more than a pipe for this video to the controller to relay to the PC where it will be processed.

C. *Hardware and Manufacturing*

1. *Mechanical Design*: The physical layout of the car was designed to take into account the various mechanical constraints of the modules described above. These constraints included placing the RFID scanner close to the bottom on the car to allow for more accurate gate reading, evenly placing IR receiver circuits around the car to allow item usage on three sides of the car, designing an anti-shake housing for the camera to prevent minor turbulence to the video feed, distributing weight evenly from the center of mass to prevent drift during normal use and placing the motor driver circuit in a location that has easy access to air-cooling for better motor control performance. Seven IR LEDs are used in a circular configuration in order to improve the range that the IR blast can travel with triggered.

2. *Analog Design*: Two circuits were designed using analog components to achieve the specifications needed for this project. These circuits were designed to minimize power consumption, maximize range of use and facilitate detection. With these design constraints in mind, the transmitter circuit shown in Fig. 7 makes use a resistor-capacitor configuration (resistors R1 and R2 and capacitor C1) to set the frequency of the NE555 Timer chip to 1kHz. This timer will drive a pulsing signal to T1. When the IR software drives T4 with a logic high, the pulsing signal is allowed to propagate to the seven IR LEDs, allowing them to blast a signal forward. C2 is simply used as a decoupling capacitor. The IR receiver circuit shown in Fig 8 makes use of a double gated input system to reduce power draw and act as a low-cost analog to digital converter by driving input signals to power rail voltages. The double-gated system allows power to only be drawn when the receiver photodiode is triggered, thus minimizing power consumption.

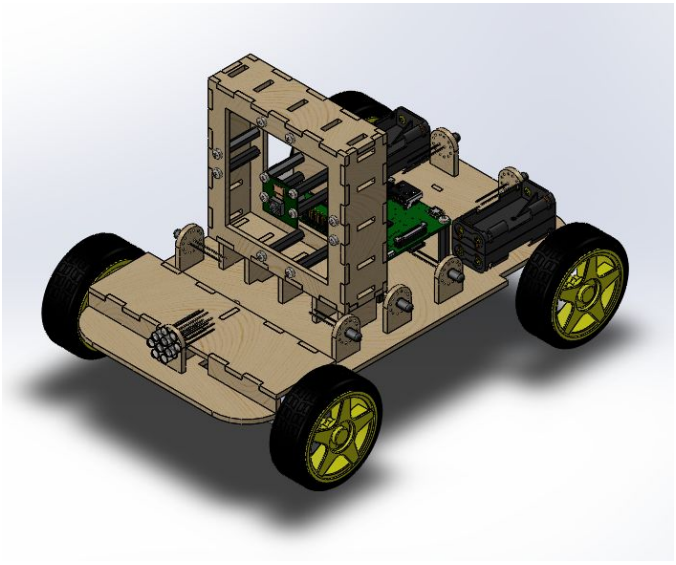


Fig. 6. 3D model for the frame of the car.

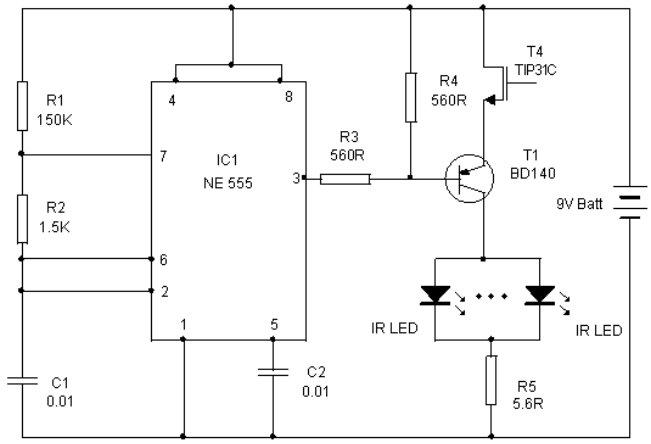


Fig. 7. Circuit diagram for IR blaster

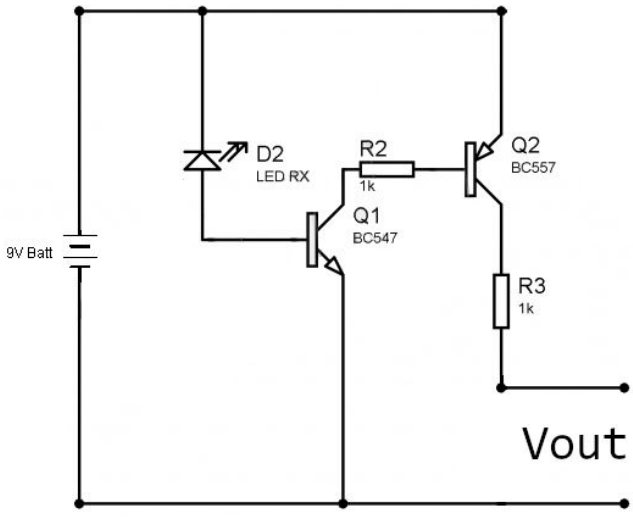


Fig. 8. Circuit Diagram for IR receiver

VI. Project Management

A. Schedule

Refer to end of report full schedule. Some reordering of tasks occurred but the general content of the schedule has not changed since the proposal.

B. Team Member Responsibilities

Broadly speaking, David is responsible for software related to the game and PCs, Sourav is responsible for software related to the Pi and its hardware, and Bujji is responsible for hardware including the physical design of the cars and circuit design, though of course we will work together when we begin integrating our components. Refer to the attached schedule for a detailed breakdown of tasks and responsibilities—David's tasks are in yellow, Sourav's in blue, and Bujji's in green.

C. Budget

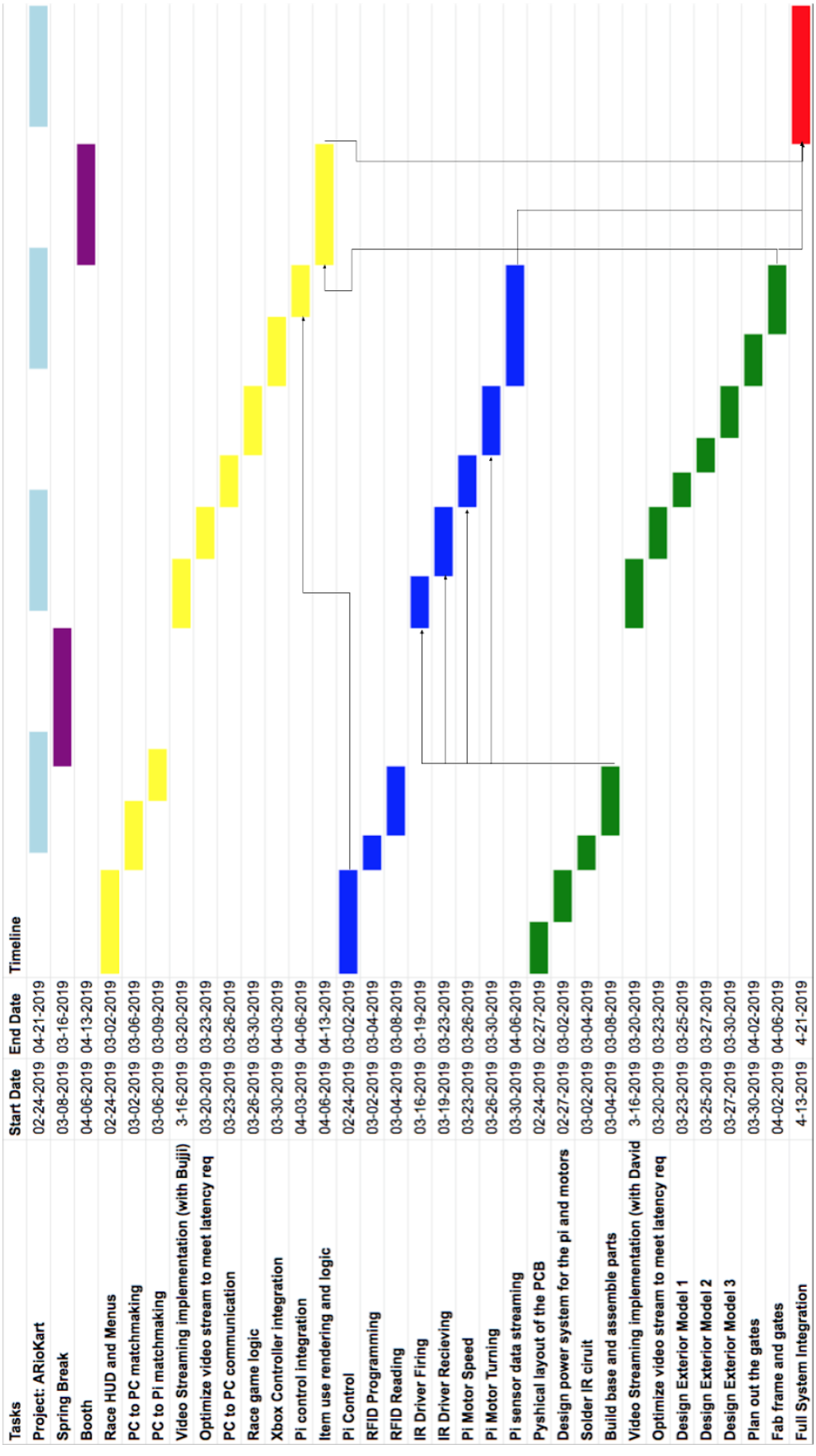
Refer to end of report for budget details and bill of materials.

D. Risk Management

1. Camera Turbulence: It was discovered that video feed stream from a small vehicle often suffers from micro-jitters. After investigating into solutions to this problem, we were unable to find affordable hardware solutions and effective real-time software to repair the blur seen in the video feed. To mitigate this risk, we have designed a low-cost camera housing that should remove minute pitch turbulence.
2. Video-Streaming Latency: A possible risk with our project is the latency in the stream of the video. Between the Raspberry Pi and the CMU Device Network, latency might be introduced simply through the artifacts of the hardware we use.
3. Budget: Since our project has a wide variety of physical components, the budget is a big risk factor. Although our current budget for all three cars and all the gates fits in the $600 allotted, we have basically no slack for broken parts or components we may not have accounted for. To mitigate this risk, we are planning to buy parts for only two cars until all other parts have been implemented. This would give us around $200 of slack to account for any problems that might arise with our parts.
4. Personal schedules: Since two of the members of our team are booth chairs, we have scheduled our working weeks around build-week. This means that there are two weeks (spring break and build-week) that members of our team will not be able to work. To mitigate this risk, we have reordered the tasks in our schedule accordingly.

References

[1] Techopedia. "What is Augmented Reality?", https://www.techopedia.com/definition/4776/augmented-reality-ar



Tasks	Start Date	End Date	Timeline
Project: ARioKart	02-24-2019	04-21-2019	
Spring Break	03-08-2019	03-16-2019	
Booth	04-06-2019	04-13-2019	
Race HUD and Menus	02-24-2019	03-02-2019	
PC to PC matchmaking	03-02-2019	03-06-2019	
PC to Pi matchmaking	03-06-2019	03-09-2019	
Video Streaming implementation (with Bujji)	3-16-2019	03-20-2019	
Optimize video stream to meet latency req	03-20-2019	03-23-2019	
PC to PC communication	03-23-2019	03-26-2019	
Race game logic	03-26-2019	03-30-2019	
Xbox Controller integration	03-30-2019	04-03-2019	
Pi control integration	04-03-2019	04-06-2019	
Item use rendering and logic	04-06-2019	04-13-2019	
Pi Control	02-24-2019	03-02-2019	
RFID Programming	03-02-2019	03-04-2019	
RFID Reading	03-04-2019	03-08-2019	
IR Driver Firing	03-16-2019	03-19-2019	
IR Driver Recieving	03-19-2019	03-23-2019	
Pi Motor Speed	03-23-2019	03-26-2019	
Pi Motor Turning	03-26-2019	03-30-2019	
Pi sensor data streaming	03-30-2019	04-06-2019	
Pyshical layout of the PCB	02-24-2019	02-27-2019	
Design power system for the pi and motors	02-27-2019	03-02-2019	
Solder IR ciruit	03-02-2019	03-04-2019	
Build base and assemble parts	03-04-2019	03-08-2019	
Video Streaming implementation (with David	3-16-2019	03-20-2019	
Optimize video stream to meet latency req	03-20-2019	03-23-2019	
Design Exterior Model 1	03-23-2019	03-25-2019	
Design Exterior Model 2	03-25-2019	03-27-2019	
Design Exterior Model 3	03-27-2019	03-30-2019	
Plan out the gates	03-30-2019	04-02-2019	
Fab frame and gates	04-02-2019	04-06-2019	
Full System Integration	4-13-2019	4-21-2019	

	No.	Part	Note	Dat	Supplier	Link	Price per p	No.	Price per ca	No.	No. per	3 Total Co	2 Total Co	3 Cat Cos	2 Cat Cost	3 Car Cos	2 Car Cost	
Control	1	Raspberry pi 3 B+			Amazon (Prime eligible)	https	$38.20	1	$38.20	3	2	$114.60	$76.40					
	2	MicroSD Card (16GI	class 10 c		Microcenter	https:	$3.49	1	$3.49	3	2	$10.47	$6.98					Decided and Ordered
	3	Camera Pi v2	Has usefu		Microcenter	https	$24.99	1	$24.99	3	2	$74.97	$49.98	$321.40	$217.56			Decided
	4	Motors	Motor(6C		Amazon (Prime eligible)	https	$18.58	2	$37.16	6	4	$111.48	$74.32					Undecided
	5	Motor Drivers	Pack of 5		Amazon (Prime eligible)	https	$9.88	1	$9.88	1	1	$9.88	$9.88					
Structure	6	Wheels (4 pieces)	2.56 inch		Amazon (Prime eligible)	https	$10.19	1	$10.19	3	2	$30.57	$20.38	$58.47	$45.28			
	7	Axle (10 pack)	200mm x		Amazon (Prime eligible)	https	$14.99	1	$14.99	1	1	$14.99	$14.99					
	8	Wood	For maki		Makerspace		$3.00	1	$3.00	3	2	$9.00	$6.00					
	9	Rubber Bands	Pack of 3		Amazon (Prime eligible)	https	$3.91	1	$3.91	1	1	$3.91	$3.91					
Power	10	Portable Battery (10000mAh)			Amazon (Prime eligible)	https	$59.90	1	$59.90	3	2	$179.70	$119.80	$117.93	$80.95	$594.09	$430.09	
		Portable Battery (6000mAh)			Amazon (Prime eligible)	https	$33.99	1	$33.99	3	2	$101.97	$67.98					
		Portable Battery (6000mAh)			Birddog Lighting	https	$29.99	1	$29.99	3	2	$89.97	$59.98					
	11	Batteries	For moto		Amazon (Prime eligible)	https	$6.99	1	$6.99	3	2	$20.97	$13.98					
	12	Battery Charger	For batte		Amazon (Prime eligible)	https	$6.99	1	$6.99	1	1	$6.99	$6.99					
RFID	13	RFID module			Amazon (Prime eligible)	https	$9.99	1	$9.99	3	2	$29.97	$19.98	$47.74	$37.75			
	14	NFC Stickers	Pack of 5		Amazon (Prime eligible)	https	$17.77	1	$17.77	1	1	$17.77	$17.77					
IR	15	IR Emitter and rece	Pack of 5		Amazon (Prime eligible)	https	$10.99	1	$10.99	1	1	$10.99	$10.99	$48.55	$48.55			
	16	Timer Chip	Pack of 3		Amazon (Prime eligible)	https	$6.58	1	$6.58	1	1	$6.58	$6.58					
	17	T1 (transmitter)	Pack of 1		Amazon (Prime eligible)	https	$6.80	1	$6.80	1	1	$6.80	$6.80					
	18	T2 (receiver)	Pack of 2		Amazon (Prime eligible)	https	$3.99	1	$3.99	2	2	$7.98	$7.98					
	19	T3 (receiver)	Pack of 2		Amazon (Prime eligible)	https	$5.70	1	$5.70	2	2	$11.40	$11.40					
	20	T4 (transmitter)	Pack of 2		Amazon (Prime eligible)	https	$4.80	1	$4.80	1	1	$4.80	$4.80					