Team Hyperion

2026 Planning and Documentation

Contents

Abstract

Hyperion is an Australian robotics team from Brisbane Boys' College, competing in the RoboCup Junior International Soccer Lightweight League and RoboCup Junior Australia Soccer Open League. The team consists of four secondary students who have developed two fully autonomous soccer robots for the upcoming 2026 RoboCup competitions. After building on two years of experience in the RoboCup Junior Soccer League in both the 2024 and 2025 seasons, our goal is to enhance the mechanical reliability, sensor accuracy, gameplay strategy and consistency of our robots.

Team Members

Hardware Role

Matthew Adams: (Mechanical & Electrical)

Matthew's robotics journey began in 2018. He quickly developed foundational skills in basic coding and problem-solving using Lego EV3 and NXT, which soon expanded to include CAD designing and modelling, as well as the ability to read electrical schematics and design and solder PCBs while understanding their functionality. His dedication to robotics is reflected in his continuous learning and commitment. Matthew has actively participated in multiple RoboCup Junior Australia (RCJA) competitions, earning impressive results such as 4th place in both State and National events, including Onstage in 2021, 3rd place in the Lightweight division at States in 2024, and 2nd place at the National competition that same year. These achievements highlight his growing expertise, passion, and competitive spirit within the robotics community.

Luke Atherton: (Mechanical & Structural)

Luke began his robotics journey in 2019, initially exploring foundational coding and Lego EV3 robotics. His skills soon broadened to include CAD design, 3D modelling, soldering, and hands-on hardware construction. Over the years, Luke has remained a dedicated and consistent presence in the RoboCup Junior Australia (RCJA) community, competing across multiple disciplines. His achievements include 2nd place in the Open Line Rescue division at the 2022 State competition and 5th nationally the same year. In 2023, he played a key role in his team's 2nd place finish at the State competition and 6th nationally in the Standard Soccer League. His continued success carried into 2024, earning 3rd place at States and 2nd place at Nationals in the Lightweight Soccer League. These results reflect Luke's growing technical expertise, collaborative mindset, and competitive drive.

Software Role

Sam Garg: (Software & Strategy)

Sam started robotics in 2018. Over the years, he has developed proficiency in a wide range of programming languages, including EV3 and Spike block coding, EV3 MicroPython, Spike Python, RobotC, and C++ within the Arduino framework. His passion and dedication to robotics have been demonstrated through consistent participation in RoboCup Junior Australia (RCJA) competitions. Notable achievements include placing 4th in both the Queensland State and National OnStage competitions in 2021, securing 2nd at State and 6th Nationally in the Standard Soccer League in 2023, and most impressively, finishing 3rd in the State Soccer League and 2nd in the National Lightweight Soccer League in 2024. These accomplishments highlight Sam's growth, technical skill, and competitive spirit in the field of robotics.

Thomas McCabe: (Software & Vision)

Thomas McCabe began his robotics journey in 2019, quickly developing a strong foundation across multiple platforms and programming environments. He has gained proficiency in languages such as Spike Block, EV3 Block, Spike MicroPython, EV3 MicroPython, and RobotC. Thomas has demonstrated consistent commitment to the field through years of participation in RoboCup Junior Australia (RCJA) competitions. His notable achievements include competing in the RCJA OnStage Challenge in 2021, advancing through RCJA Secondary Rescue in 2022 and 2023, and most recently, earning a 3rd place finish at the State level in the RCJA Soccer Standard League in 2024. These milestones reflect not only Thomas's technical progression but also his resilience and enthusiasm for robotics.

Software Planning

Assignments

Aspects	Assigned To	Individual Order	Overall Order	Status
Bluetooth	Sam	3	6	Complete (26/10/25)
Camera (Teensy Side)	Sam	2	5	Complete (30/10/25)
Camera (OpenMV Side)	Tom	3	4	Complete (29/10/25)
Drive System	N/A (Prev.)	N/A (Prev.)	N/A (Prev.)	Complete (Prev.)
Light System	Tom	2	3	

PID	N/A (Prev.)	N/A (Prev.)	N/A (Prev.)	Complete (Prev.)
TSSP System	Tom	1	1	Complete (29/10/25)
Timer	N/A (Prev.)	N/A (Prev.)	N/A (Prev.)	Complete (Prev.)
Voltage Divider(s)	N/A (Prev.)	N/A (Prev.)	N/A (Prev.)	Complete (Prev.)
Compass	N/A (Prev.)	N/A (Prev.)	N/A (Prev.)	Complete (Prev.)
Kicker Mechanics	Sam	4	7	Complete (26/10/25)
Dribbler Mechanics	Tom	5	9	
Kicker Strategy	Tom	4	8	
Dribbler Strategy	Sam	5	10	Awaiting Mechanics Side
Movement Strategy	Sam	1	2	Complete (21/10/25)
Common	Shared	Shared	Shared	Shared
Configuration	Shared	Shared	Shared	Shared
Pins	Shared	Shared	Shared	Shared

General Conventions

Variables – Camel Case

Class Names – Camel Case with a capital starting letter

Function and Struct Names – lower case separated by underscores

1. TSSP System – Thomas McCabe

Define Problem

In order for the robot to gain possession of the ball and score, it must first be able to determine the direction of the ball relative to itself.

Key Question:
How can the direction of the ball relative to the robot be accurately determined?

Background Research

There are several possible methods for detecting and determining the relative direction of the ball.
Since the competition ball is orange, one option is to use a vision-based system such

as the OpenMV H7 camera. This camera can identify coloured objects — for example, yellow and blue goals, and an orange ball — allowing the robot to estimate the ball's position. However, this approach has limitations, such as the camera's frame rate and potential loss of visual contact if the ball is obscured.

Alternatively, the competition ball emits infrared (IR) light, which can be detected using TSSP sensors. Infrared light has the advantage of being less affected bytea visual obstructions, making this method generally more reliable for continuous tracking.

Requirements

- The offset values of each TSSP sensor must be easy to adjust.
- The system must provide a consistent and reliable measure of ball signal strength.
- The program must execute quickly and efficiently on the Teensy microcontroller.

Plan for Solution

The proposed solution involves reading data from each TSSP sensor 255 times and summing the digital outputs (0 or 1) to calculate a total value for each sensor. The sensor with the highest total value indicates the direction of the ball, expressed in degrees relative to the robot.

Functions

Name	Description
init_tssp	To set up the pins and their types
read_tssp	To read all the tssp and find ball dir and str
get_ball_dir	Returns ball dir
get_ball_str	Returns ball str

2. Movement Strategy – Sam Garg

Attacker

Define Problem

To not constantly hit the ball towards the defending goal, the robot must have a movement strategy to effectively get behind the ball and move it towards the opponent's goal.

What kind of attacking movement strategy should the robot have?

Background Research

There are many different types of orbits/movement strategies, which can include:

Orbit Type	What is it?	Positives	Negatives
Cases Orbit	A case's orbit is a set movement angle when the ball angle is within a certain range of angles.	• Easy to set up. • Predictable movement (always behave the same within each case). • Low processing demand.	• Lengthy and tedious tuning. • Jerky movement. • Does not adapt well to unexpected ball positions or fast movement. • Hard to maintain smooth transitions between cases.
Fixed Offset Orbit	A fixed offset orbit is a set constant that adds or subtracts from the ball's direction.	• Easy to set up. • Easy to tune. • Consistent and simple. • Low processing demand.	• Orbit can be very wide or too close at times, with no variance. • Lack's flexibility • Struggles when the ball moves quickly or unpredictably.
Exponential Polynomial Orbit	An exponential/polynomial orbit is a constantly changing offset value that is subtracted or added to the ball's direction.	• Smooth movement. • Tightens naturally near the ball. • Looks natural and efficient in play-like situations. • Handles varied speed sand angles better than fixed systems.	• Complex to set up. • Harder to predict behaviour/interpret • Medium processing demand.
PID Orbit	A PID Orbit has a vertical and horizontal vector; it uses these vectors to wrap around the ball.	• Very smooth and adaptive movement around the ball • Continuously fixes itself based on readings; maintains stability. • Adjustable responsiveness. It can be very aggressive if needed.	• Very complex to set up. • Very complex to tune. • Can overcorrect or oscillate if not tuned properly. • Very high processing demand. • Relies on ball strength and distance; cannot function without.

Requirements

The orbit movement strategy must:

- Be easy to tune and adjust.
- Position the robot efficiently behind the ball.
- Prevent unwanted contact with the ball (from sides or rear)
- Adapt to different ball speeds and movement patterns.
- How low to medium processing demand (to optimise the loop speed)
- Be stable and predictable
- React quickly to sensor data
- Maintain smooth curvature in its orbit path
- Allow for scalable complexity (make it more advanced)
- Minimize overshooting

Plan for Solution

Based on the background research and the outlined requirements, an **Exponential Orbit** strategy will be implemented in the program. This method will rely on a single constant to tune the overall orbit behaviour.

1. A **modified ball direction** will be calculated, ranging from -180° to 180°. This allows both sides to be mirrored, ensuring the robot produces symmetrical movement on either side of the ball.

2. A **movement scalar** will be determined based on the ball's position. This value will range between 0 and 1 and follow an exponential curve, allowing for smoother and more responsive movement as the ball gets closer.

3. A **movement offset** will be generated, increasing exponentially based on how far off-centre the ball is. This creates a smooth orbiting path, with the offset capped at 90° to prevent the robot from turning completely sideways. The offset will then be added or subtracted from the ball's direction, depending on which side the ball is on, to determine the final movement direction.

This plan will be integrated into the TSSP library as a separate function named orbit().

Defender

Define Problem

The opponents' robots will aim to score against our goal. To stop them from scoring when the attacker cannot get back around the ball in time, it is wise to have a robot hang back, defending the goal.

How should the Defender move, and what algorithms can be used to calculate the Defender's movement?

Background Research

Defending can be done in many ways. The following table describes the different methods that can be used to program the defender.

Method	What is it?	Positives	Negatives
Distance Constrained Defence	The robot orbits around the ball but stops moving forward once it reaches a set distance from the goal.	• Not commonly seen and provides a level of difference • Higher chance of scoring compared to staying in deep defence • Helps maintain goal positioning between the ball and goal.	• If opponents have a strong kicker, the robot may not orbit back fast enough to block the ball. • Can leave the goal open if timing or distance calibration is off.
Reactive Zone Defence	The robot stays between the goal and the centre of the field, moving or orbiting only when the ball enters its defensive zone.	• Conserves energy and reduces unnecessary movement. • Provides reliable defensive coverage when the ball approaches. • Easier to tune than complex systems.	• May take too long to react to fast-moving balls. • Can struggle against opponents with quick gameplay.
Floating Defender using Vectors	The robot keeps a constant distance from both the ball and the goal, always positioning itself between them to be ready for quick attacks or strong defence.	• Offers balanced positioning for both offence and defence. • Enables faster transitions between attacking and defending. • Maintains optimal alignment with the ball and goal.	• Requires precise vector calculations and tuning. • Can become unstable if the ball moves unpredictably and it is not tuned correctly.
Defender using Vectors	The robot moves sideways with the ball while saying a set distance away from the goal.	• Simple and stable defensive approach • Effective for cutting off shots to the side of the goal with goal tracking. • Easier to implement and maintain.	• Less responsive to central attacks. • Doesn't adapt to ball position relative to the goal (distance wise). • Can create open angles for shots if not tuned properly.

Requirements

- Maintain Goal Coverage
- Efficient Movement (minimise unnecessary movement)
- Fast Reaction Time
- Stable Positioning
- Accurate Distance Control
- Smooth & Visible Switching (You must be able to see a visible change between attacker and defender strategies).
- Algorithm Flexibility & Tuning

Plan for Solution

Given the positives and negatives of the chosen solutions, a floating defender using vectors will be picked. This is because the robot will be positioned between the ball and the goal, which maintains coverage. It can be easily tuned to minimise unnecessary movement and can be stable when velocities are limited, and dampening is added.

- When the ball is not on the field, the robot will centre relative to the goal the minimum distance away (this will be a define to tune).
- When the ball is on the field, the robot will use a tuned horizontal vector to position itself in front of the ball.
- The robot will actively face away from the goal so that when the robot moves towards the ball it faces between the ball and goal.
- A maximum and minimum goal distance will be defined to ensure that the robot does not move too excessively towards or away from the defending goal.
- A scaling constant will be defined between the ball strength and the goal distance, attempting to make them similar values. Whichever value is smaller will be multiplied by the scaling constant.

The logic for this solution will be placed inside the main file as it requires multiple objects.

3. Light System – Tom McCabe

Define Problem

In order to for the robot to stay in play it must not go outside of the fields white line.

How to locate the direction of a white line?

Background Research

For a robot to remain within the boundaries of a playing field, it must be able to detect and respond to the white line marking the edges. This requires an effective line detection system capable of distinguishing the white line from the surrounding field surface, which is often darker in color (e.g., green turf or black mat).

A common approach to detecting white lines involves light sensors, such as photoresistors, photodiodes, or phototransistors, in combination with light-emitting diodes (LEDs). These sensors work by measuring the intensity of reflected light. When a surface is illuminated by an LED, a white surface reflects significantly more light than a darker surface. The phototransistor converts the reflected light into an electrical signal, allowing the robot to detect the contrast between the line and the field.

Using a ring of photo transistors and LEDs arranged around the base of the robot provides 360° coverage, enabling the robot to detect the white line from any direction. This setup helps determine not only the presence of the line but also its direction relative to the robot. When one or more sensors detect a strong reflection (indicating a white line), the robot's control system can infer which direction the boundary lies and take corrective action to stay in play.

Alternative methods for line detection include:

- Infrared sensors: Use IR light to detect reflectivity differences between surfaces.
- Camera-based systems: Employ computer vision to detect line patterns and colors.
- Color sensors: Identify specific color values to distinguish field markings.

However, the photo transistor and LED ring solution offers a lightweight, low-cost, and fast-response method well-suited for real-time boundary detection in small autonomous robots.

Requirements

- To be able to detect the line
- To be able to remeber the direction even if your not on it
- To be able to reconise if youve gone over the line

Plan for Solution

By creating clusters of sensors pass the white threshold then finding the angle of the clusters centers and averaging them.

Functions

Name	Description
init_lsb	To set up the pins and their types
calculate_line_direction	To read all the photo tran and find line dir
get_line_state	Returns line state
get_line_dir	Returns line dir

4. Camera Library (OpenMV Side) – Tom McCabe

Define Problem

To locate the position of the ball and goal, and send them to the teensy.

How do I locate the ball and goals using an OpenMV Camera.

Background Research

Open Mv is a software that allows you to use an open mv camera and use python to interrupt the image every frame this info of the position of the goals and the ball using their colour as the way to locate them.

- Use blob tracking
- Other method
- How do I send over data

Requirements

For the teensy to receive the (x,y) positions of the goals and the ball in order to now their relative direction.

Plan for Solution

The plan is to use the find blobs function in open mv to locate blobs of colour that represent the ball (orange) and the goals (blue and yellow), then use the uart system to sent there pixel coords over to the teensy to be interrupted into a relative direction.

5. Camera Library (Teensy Side) – Sam Garg

Define Problem

The values for goal position and ball position are sent over UART from the OpenMV Camera. The teensy must communicate with the OpenMV to find the position of the goal and ball.

How should the teensy manipulate and receive these values off UART to calculate the position of the goals and ball?

Background Research

The UART values will be sent over in this exact order:

- Camera Start Packet 1
- Camera Start Packet 2
- Yellow X
- Yellow Y

- Blue X
- Blue Y
- Ball X
- Ball Y

To receive these values off UART, Serial functions must be used. An optimal way to use these functions can be seen in Source 5 of this document.

However, once these values are received, certain equations should be used to calculate the exact angle and distance of each of the goals and the ball.

Type	Equation
Angle Finder	$f(x,y) = (90 - \tan^{-1}\frac{y}{x})\ mod\ 360$ This function allows us to accurately output an angle when given an x and y value.
Distance Finder	$f(x,y) = \sqrt{x^2 + y^2}$ This function allows us to accurately output a distance when given an x and y value.

Requirements

- Solution must account for noise across UART channel.
- Solution must accurately receive values from the OpenMV through UART.
- Solution must calculate angle and distance of blue and yellow goals as well as orange ball.
- Solution must assign the attacking and defending goal depending on assigned attacking goal.

Plan for Solution

The solution will have many varying functions; the below table outlines a plan for each function.

Function	Perameters	Plan
init (void)	None.	Initialises the camera serial for use (115200 baud rate).
update (void)	attackingGoal – Which goal is the robot attacking? Blue is true, yellow is false.	1. If the serial has more than 8 bytes available, continue to the next step. 2. Check if the first 2 start bytes are correct, if they are continue to the next step. 3. Read the serial 6 consecutive times and assign these values to associated variables. 4. Subtract any values needed from each variable based on what is

		done in the OpenMV code (as negative values cannot be sent over UART). 5. Assign the attacking and defending goals based on the attackingGoal perameter.
calculate_angle (float)	x y	Uses: $f(x,y) = (90 - \tan^{-1}\frac{y}{x}) \; mod \; 360$
calculate_distance (float)	x y	Uses: $f(x,y) = \sqrt{x^2 + y^2}$
get_attack_goal_angle (float)	None.	Self-explanatory get event.
get_attack_goal_distance (float)	None.	Self-explanatory get event.
get_defend_goal_angle (float)	None.	Self-explanatory get event.
get_defend_goal_distance (float)	None.	Self-explanatory get event.
get_ball_angle (float)	None.	Self-explanatory get event.
get_ball_distance (float)	None.	Self-explanatory get event.

6. Bluetooth – Sam Garg

Strategy

Define Problem

To not crowd the ball and to split roles on the field, the robot will need to have a strategy as to deciding which one is the attacker, and which one is the defender.

What is the condition(s) for deciding which robot is attacking and which is defending?

Background Research

The following table describes the different conditions/methods that can be used to decide the attacker/defender roles in a game situation.

Method	What is it?	Positives	Negatives	Combine with
Strongest Ball Strength	The robot with the strongest ball strength becomes the attacker. The other becomes the defender by default.	• Simple and reliable indicator of which robot is closest to the ball. • Fast decision-making with minimal data processing • Reduces confusion in determining possession. • Works without more complex components (e.g. Camera).	• Can cause rapid switching if both robots have similar ball readings. • Doesn't consider field position (may send both robots too far forward). • Could result in collisions near the ball if not paired with direction/goal checks.	Ball Direction Attack Cone. Distance-from-own-goal Fallback
Ball Direction Attack Cone	If a robot's shows the ball within a forward cone (e.g. ±30 degrees of its heading) and the ball strength is similar to that of the other robot, the robot should be the attacker. The other robot becomes the defender by default.	• Ensures the attacker is properly oriented toward the ball. • Reduces risk of sideways or backwards approaches. • Promotes smoother, more strategic ball control and chasing behaviour. • Avoids conflicts where both robots see the ball from opposite sides.	• Relies on more accurate heading readings, errors can cause false alignment. • May ignore a closer robot if its angle is slightly off. • Slower to decide if angle calculations are noisy.	Strongest Ball Strength Goal Alignment
Goal Alignment	If a robot can see the opponent goal and the goal heading is generally forward from it, prefer that robot as	• Encourages intelligent play by choosing robots that are well-positioned to score.	• Goal detection can be inconsistent under lighting changes.	Ball Direction Attack Cone

	attacker. The other becomes the defender by default.	• Prevents attackers moving in the wrong direction. • Improves orientation consistency. • Increases scoring efficiency once ball possession is achieved.	• May fail if the goal is blocked or temporarily useless. • Doesn't help much during defence phases or when ball is near your own goal.	Distance-from-own-goal Fallback
Distance-from-own-goal Fallback	Whichever robot is closer to the goal defaults to the defender. If both robots are near the defending goal, use other conditions (such as strongest ball strength) to determine roles.	• Ensures at least one robot is always defending the goal. • Prevents both robots from rushing too far up the field. • Provides structure and balanced team positioning. • Easy to determine using compass and estimated positioning using camera.	• Position may be slightly inaccurate/inconsistent at times. • May cause excessive hesitation if both robots stay defensive. • Can conflict with goal alignment if the "defender" also has a strong shot.	Strongest Ball Strength Goal Alignment
Battery Priority	If one robot's battery/temperature/health is below a safe threshold, deprioritize it for attacking tasks.	• Reduces risk of inconsistent sensor values or slowed gameplay on one robot compared to the other. • Reduces heat buildup and motor strain on weaker robot. • Prevents performance drops during key moments and scoring opportunities.	• Requires constant battery monitoring and sending this via Bluetooth. • Can cause uneven role assignments if one robot's battery remains low for the entire match. • Doesn't consider game context (e.g. what is happening in the game)	Strongest Ball Strength Distance-from-own-goal Fallback

Requirements

- Consistent Role Assignment Logic
- Scalability
- Simplicity
- Small Number of Aspects/Values Required (to prevent the number of things being sent over Bluetooth)
- Synchronization (One robot must be attacking and one must be always defending)

Plan for Solution

A variety of the above brainstormed strategies will be incorporated into the final solution.

The robots will share key data values:

- Ball Strength
- Ball Direction
- Goal Angle (Attacking Goal)
- Goal Distance (Defending Goal)
- Battery Level

Using these values, the robot will decide whether it is an attacker or defender based on prioritised on the following set of conditions in a hierarchy.

1. **Compare Battery Levels**
   If one robot's battery is low, it automatically becomes the **defender**.
2. **Compare Ball Strength**
   The robot with the higher ball strength becomes the **attacker**, unless Step 3 or 4 overrides it.
3. **Check Ball Direction (Attack Cone)**
   If one robot sees the ball within a forward cone (±30° of its heading) and the other doesn't, that robot becomes the **attacker**.
4. **Goal Alignment**
   If a robot is oriented toward the opponent's goal, it remains or becomes the **attacker**.
5. **Use distance-from-own-goal fallback**
   If both robots meet similar attack criteria, the one closer to the defending goal becomes the **defender**.

With these conditions, the behaviour of each robot in each role should appear as:

- **Attacker:** Moves toward the ball, maintains alignment with the opponent's goal, and attempts to shoot when close enough.

- **Defender:** Holds position closer to its own goal, tracking both he ball and the attacker's location to intercept or block if the ball approaches the goal area.

In addition to this, to ensure that there is a "fail-safe" to this logic, should Bluetooth data be disconnected, lost, or unclear, roles default to **Defender** logic. Having two robots in the back half of the field is better than leaving the goal open and exposed.

Sending and Receiving

Define Problem

To send the values that decide strategies across both robots, a process must be defined.

What process will this be?

Background Research

How receiving should work: (Source 1 of this Document)

1. First check if there is anything that is available on the serial (Serial.available()). If there is anything available, continue to the next step, if not keep checking.
2. Serial.read() to get this value and delete it from the frame buffer.

This is how values can be received from the other HC-05 Bluetooth module.

What we can take away from this source: How to send and receive using HC-05 Bluetooth module.

Noise across serial lines: (Source 2 of this Document)

In the above source, we can see that the user is experiencing issues with the HC-05 Bluetooth module sending odd values here and there. This is called noise. To prevent noise, two start bytes can be used to ensure that the values being read are inside the packet that we want to read.

E.g. Start bytes can be 255 and 255 in a row as no sensor input will output 255 twice in a row.

What we can take away from this source: We should include start bytes in our sending and receiving codes to find the start of a packet of values.

Sending: (Source 3 of this Document)

In the above source, we can see how to send values from a HC-05. This can be a simple Serial.write() function to the specific serial. Ensure the 2 start bytes are sent first before any other values are sent.

HC-05 Packet Support: (Source 4 of this Document)

The HC-05 can support up to 38400 bits per second (38400 baud rate).

Name	Minimum Required Data Size
Byte1	1 byte (8 bits)
Byte2	1 byte (8 bits)
InfoByte	1 byte (2 bits)
Ball Strength	1 byte (8 bits)
Ball Direction	2 bytes (9 bits)
Goal Angle	2 bytes (9 bits)
Goal Distance	1 byte (8 bits)
Battery Level	1 byte (8 bits)

Therefore, the data size of the packet will be a minimum of **9 bytes**.

Calculate the packet size:

$$10\ bytes = 10 \times 8\ bits = 80\ bits$$

Calculate the maximum number of times it can send per second:

$$\frac{38400\ bits/sec}{80\ bits} = 480\ times\ per\ second$$

Calculate how frequent this is (microseconds):

$$\frac{1000\ milliseconds}{480\ sends\ per\ second} = 2.083\ milliseconds\ per\ send$$

For the HC-05 to be running at maximum capacity, we should send our packet every 2.083 milliseconds. However, to not overload the HC-05 and teensy module, we should run the HC-05 at a maximum of 10% of its performance. Therefore,

$$480\ times\ per\ second \times 10\% = 48\ times\ per\ second$$

$$\frac{1000\ milliseconds}{48\ times\ per\ second} = 20.83\ milliseconds\ per\ send$$

Algorithm based on performance:

$$t_{send}(ms) = \frac{208333}{P}$$

$$P = performance\ rate$$

$$ms = microseconds$$

What we can take away from this source: For the HC-05 to be running at 10% capacity, a packet should be sent every 20.83 milliseconds.

Requirements

- Accurate Data Transmission
- Start Byte Packet Framing

- Error Resistance and Noise Handling (Ignore any data received before the start bytes)
- Defined Packet Structure
- Efficient Transmission Timing (Controlled Intervals)
- Fail-safe Defaults
- Compact Data Format (Compressed Values)

Plan for Solution

To ensure consistent and reliable Bluetooth communication between both robots, the following sending and receiving system will be implemented.

The defined data packet structure will be as follows:

- [Byte1, Byte2, InfoByte, BallStrength, BallDirection, GoalAngle, GoalDistance, BatteryLevel]

As discovered previously, this packet will be sent every 20.33 milliseconds to run the HC-05 at 10% performance. However, this can be tested and tuned using the function:

$$t_{send}(ms) = \frac{208333}{P}$$

There will be a few main functions inside the library, the following table describes these.

Function	Brief Explanation	Plan
Init (void)	Prepares the library for usage in the setup.	- Begin serial communication at the defined baud rate. (38400) - Reset and start all timers.

update (void)	Updates the whole object in a loop.	- Update local self-data with current sensor readings.
		- Call send() when the timer expires and read whenever data is available.
		- Determine if the Bluetooth connection is alive using a timer.
		- Use hierarchal decision logic to assign self.role and manage switching (the details can be seen in the strategy section for Bluetooth).
		- Prevent Rapid oscillations with a cooldown timer.
		- Handle fail-safe defaults.
		  o If Bluetooth disconnects à force defender.
		  o If both robots can't see ball à pause switching.
		- Include debug output for testing.

send (void)	Writes the data to the other HC-05 module.	- Send two start bytes (255, 255) - Send all fields in compressed format (defined above in the data packet structure) - Converts floats into scaled integers for consistent transmission. - Reset a timer after transmission.
read (void)	Receives the data from the other HC-05 module.	- Continuously check if enough bytes are available to form a full packet - Look for two consecutive start bytes (255, 255) - Once found, read remaining bytes in defined order and decode them into other data fields. - Update a timer each time a valid packet is received
get_role (Get Event) (bool)	Outputs the role of the local robot.	- Return the Boolean value of self.role - No calculations occur here.

7. Kicker Mechanics – Sam Garg

Define Problem

To use our kicker, we must create a function which allows us to 'fire' the kicker. There are some constraints as to when we should fire our kicker, and how this can be done easily through the code.

What are the constraints for the kicker software wise, and how can this be implemented via code?

Background Research

Kicking too frequently:

- Teams in the past have reported kicking too frequently. To fix this, we will implement a timer which will ensure that the kicker does not fire too frequently.

Kicking before enough voltage is generated:

- To ensure that the kicker does not fire before enough voltage has been generated, a voltage divider will be read using the voltage divider library and then passed into a function.

How to Kick:

- Writing high to a digital pin which enables power to the kicker.

Requirements

- The kicker must only fire when the capacitor voltage is above a defined threshold.
- The kicker must have a cooldown timer to prevent it from firing too frequently.
- The kicker must activate through a controlled digital output (e.g. writing HIGH to a specific pin for a short, defined pulse duration)
- The kicker logic must be efficient and non-blocking (no delay() calls that pause other functions)
- The software must protect against accidental double-triggering caused by loop timing or communication errors.

Plan for Solution

When combining the above information, a clear plan for the solution can be found. Below are the two functions that will be inside the library:

Function	Parameters	Functionality
init (void)	None.	Initialises the library for use: ➢ Define the pin mode for the kicker pins. ➢ Reset timers.

fire (void)	Capacitor Voltage (voltage divider)	➢ Continuously read the capacitor voltage using the voltage divider library (value fed into function) ➢ Store the voltage value and compare it against a defined threshold to ensure efficient charge. ➢ Use a timer system to track cooldown time between kicks and prevent frequent firing. ➢ When voltage and timer conditions are met, briefly write HIGH to the kicker's digital output pin to activate the solenoid. ➢ After a short, defined pulse (100-200 ms), set the pin LOW to deactivate the kicker. ➢ Reset the cooldown timer immediately after each kick to manage timing between activations.

8. Kicker Strategy – Tom McCabe

Define Problem

In order to utilise the kicker effectively it must only be used when it will aid in scoring.

Background Research

to use the kicker effectively it must only be used when it aids in scoring, this means that the kicker should not fire unless the robot has possession of the ball. This can be checked by using a light gate to know when the robot has possession. The kicker also should not be fired unless it is being aimed at the goal as well as it shouldn't kick on its side of the field.

Requirements

- To be able to more effectively score goals
- To have better accuracy

Plan for Solution

When on the enemies side of the field, as well as facing the goal, and the kicker will be fired into the left or right side of the goals in order to make the shot harder to defend.

9. Dribbler Mechanics – Tom McCabe

Define Problem

In order to maintain control of the ball a motor will be used to pull the ball into the robots capture zone.

How to utilize a motor to maintain control of the ball?

Background Research

To maintain control of the ball, the robot must be able to detect when the ball enters its capture zone and then activate a motor to pull it in securely. This can be achieved through the use of ball strength combined with a motor control system.

Once the software detects that the ball strength is high as well as the ball being in front of the robot, it can trigger the motor to spin forward, pulling the ball into the robot's capture zone. This process can be managed through a simple state-based control system in software. The logic typically follows these steps:

1. Idle State:
   The robot continuously monitors the ball strength. The motor remains off while waiting for the ball to be detected.
2. Capture State:
   When the light gate is interrupted, the software identifies that the ball is entering. The motor is then activated to rotate forward, drawing the ball inward.

The system can be further refined with additional software features such as:

- Timing control to prevent the motor from running too long.
- Current monitoring to detect jams or overloads.
- signal filtering to ignore brief false triggers from ambient light or noise.

By combining ball strength for ball detection with a motor control algorithm, the robot can maintain consistent and efficient control of the ball. This software-driven approach ensures reliable performance with minimal mechanical complexity, enabling smooth transitions between detecting and capturing the ball during gameplay.

Requirements

- To be vertical and useable in all situations
- To be able to be used with a kicker
- To be able to change the dribbler speed and direction

Plan for Solution

The plan is to use ball strength to detect the ball then have the dribbler attached to the motor be able to change its speed and direction independently.

Functions

Init_dribbler	Init the motors
run_dribbler	Sets the speed and direction of the dribbler

10. Dribbler Strategy – Sam Garg

Define Problem

If the dribbler runs successfully when the ball is near, should the robot's strategy be different?

Background Research

Strategy	Description	Pros	Cons
Ball Snatching	Move to face the ball, dribble it, face towards the opponent's goal and kick.	• Fast reaction time allows quick attacks or interceptions. • Effective for sudden ball turnovers and counterattacks. • Simple to implement and requires less complex positioning.	• High risk of losing ball control after initial contact. • Often misaligned with the goal after grabbing the ball. • Inefficient against robots with stronger dribblers or better defense. • May waste energy due to constant acceleration and direction changes.
Orbit and dribble	Orbit behind the ball, dribble it, face towards the opponent's goal and kick.	• Maintains smoother, more consistent ball control. • Positions the robot behind the ball for accurate goal alignment. • More strategic — allows planning and aiming before shooting. • Reduces risk of losing possession to opponents. • Works well with sensors and camera-based tracking for precision.	• Slightly slower to reach the goal compared to direct approaches. • May hesitate or stall if the ball's position changes rapidly.

Requirements

- The robot must maintain stable ball control while moving.
- The robot must avoid unnecessary energy use and maintain efficient movement.
- The robot must recover control quickly if the ball moves unexpectedly.
- The robot's code must be consistent, reliable, and compatible with competition rules.
- The robot's strategy must maximize scoring opportunities while minimizing loss of possession.

Plan for Solution

Based on the background research and the outlined requirements, an Orbit and Dribble strategy will be implemented to maintain stable ball control while approaching the goal. This strategy positions the robot behind the ball and gradually adjusts its path, allowing smoother movement, precise goal alignment, and reduced risk of losing possession.

1. The robot will regularly orbit around the ball, as discussed in the attacker movement strategy section.

2. When the ball is near, the dribbler will activate maintaining possession of the ball.

3. The robot will angle itself towards the goal and will move forward until it is a certain distance away.

4. Once it is close enough, the robot will kick the ball into the goal.

Hardware Planning

Major Improvements / Changes:

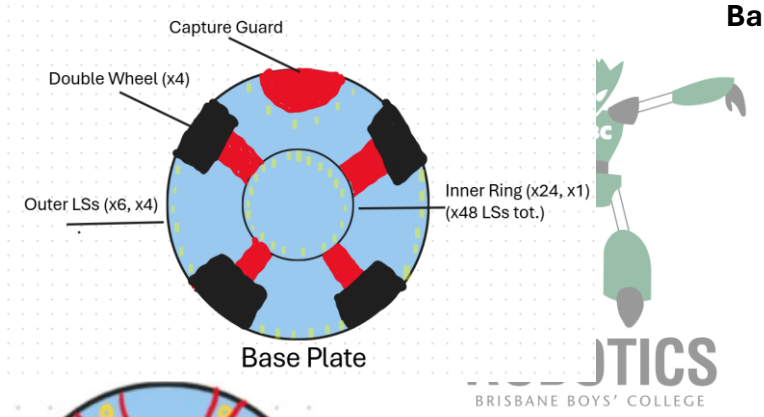
★ (Priority)	Improvement / Change	Problem / Rationale	Improvement Details / Changes Made	Expected Outcome / Benefit
★★★★	**CNC Mirror**	• Current printed mirror has distortion and rough reflection. • Hard to accurately calculate pixel distance for localisation. • 3D-printed surface inconsistent under lighting.	• **CNC machine the mirror** from aluminium for perfect curvature and reflection. • Use proper hyperbola calculation ($y^2 = kx$) to match OpenMV field of view. • Add a machined mounting lip for consistent alignment. • Polish finish for minimal reflection noise.	• **Less distortion and clearer camera vision.** • Easier pixel-to-angle localisation. • More reliable and repeatable goal detection.
★★★★★	**Kicker (Solenoid Based)**	• Robot relied on momentum for ball release — slow and inconsistent. • No ability to shoot accurately or from distance.	• Added **kicker PCB** (XL6009 voltage booster + capacitor + solenoid). • Integrated **lightgate sensor** to detect when ball is in position. • Optimised solenoid position for ball clearance and safe wiring. • Improved frame mount to absorb impact.	• **Can shoot and pass accurately.** • More consistent scoring and strategy options. • Reliable ball release every time.
★★	**Dribbler (Optional)**	• Would improve ball control but adds mechanical and wiring complexity. • Not essential for current strategies.	• Designed lightweight single-motor dribbler (low torque). • Mount compatible with kicker frame. • Test implementation only if time allows.	• Smoother control while moving. • Helps align shots, optional extra.
★★★★	**Lower Centre of Mass (COM)**	• Slightly top-heavy → wobble during acceleration. • IR ring position limited by battery height.	• **Move motors further outward**, thinning outer shell. • **Flatten battery** to lower position. • Drop IR ring lower (no longer blocked diagonally). • Keep base + mid aluminium plates to maintain low COM. • Use **honeycombing** if weight becomes an issue.	• **More stable and balanced driving.** • Lower IR ring = better signal consistency. • Improved turning accuracy and reduced wobble.
★★★	**Traction / Double Wheels**	• Current single wheels lose rollers easily. • Smaller contact area → less stability. • Narrow base reduces grip.	• **Test double wheel setup** for wider base and extra grip. • Recut rollers from higher-quality silicone tubing. • Verify 3-roller contact at all times. • Slightly widen frame if needed.	• **Better grip and traction.** • Less roller loss. • Smoother driving over uneven surfaces.
★★★★	**PCB Reliability Improvements**	• Random compass disconnects and fuse issues. • Power and sensor reliability inconsistent.	• **Thicker traces** on high-current lines. • Improved solder quality and connector selection. • Replace compass with **BNO085** (more reliable). • Add mechanical support under compass module. • Option for dual-compass redundancy if needed.	• **Far fewer electrical faults.** • Easier debugging and higher match reliability.
★★★	**Battery Connection Update**	• Battery plug occasionally loosened mid-match. • Hard to swap quickly.	• **Locking XT60 connector** or stronger socket. • Add 3D-printed retention clip / strain relief. • Clean up cable routing to make swapping easier.	• More secure connection. • **Faster battery changes.** • Reduced power dropouts.
★★★	**Baseplate Height Adjustment (for LSB Guard)**	• Old baseplate scraped the ground — no room for LSB guard. • Guard removed to avoid contact.	• **Raise baseplate slightly** for clearance. • Add mounting space for LSB guard. • Reassess ground clearance after lowering COM.	• Can refit **LSB guard**. • Prevents scraping. • Protects underside from damage.
★★★	**Comms Module Integration (Main PCB)**	• Had to attach communication board manually at comps → messy wiring and slow setup.	• **Add dedicated pins** for comms module (HC-05 / HC-06) on main PCB. • Label headers for plug-and-play. • Reserve small PCB space for module.	• Cleaner wiring. • Faster comp setup. • Reduced connection errors.
★★★½	**Capture Guard Redesign**	• New ball is slightly different size and weight. • Harder to trap reliably.	• Resize capture guard to fit new ball snugly. • Add **soft inner layer (foam / sponge)** for better grip.	• **Better hold and control on ball.** • More consistent capture and release.

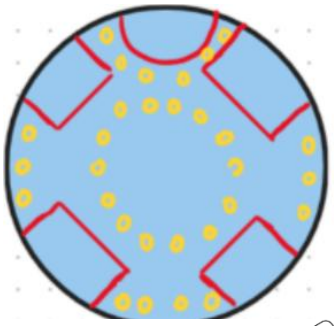
⭐⭐⭐⭐	**Connection Reliability (BNO, Motors, PCB)**	• Ball bounces off solid surface. • Loose headers and motor plugs caused random cut-outs mid-match.	• Allow kicker clearance. • Optional lightgate sensor mount. • Use **locking connectors** for BNO + motor cables. • Re-solder joints carefully. • Add cable clips or tie-downs for strain relief. • Small dab of glue on key headers.	• **Fewer mid-match disconnections.** • Stronger electrical reliability overall.
⭐⭐	**Other (LSB, blank, blank, blank, blank)**	• Full outer ring doesn't fit with new frame changes, inner only is limited	Use **outer sections + inner ring**: (Inner ring → simpler maths and reliable for line avoidance) (outer sections → detect line contact for code cases (e.g. slower or stop))	Keeps LSB functionality within new layout: (Inner = reliability) (outer = code triggers.)

Notes:

- If total weight increases with kicker + mirror → honeycomb aluminium base/mid plates to cut mass.
- Focus areas for early build phase: Kicker, CNC mirror, PCB durability, lower COM.
- Dribbler only added if core systems are stable and time allows.

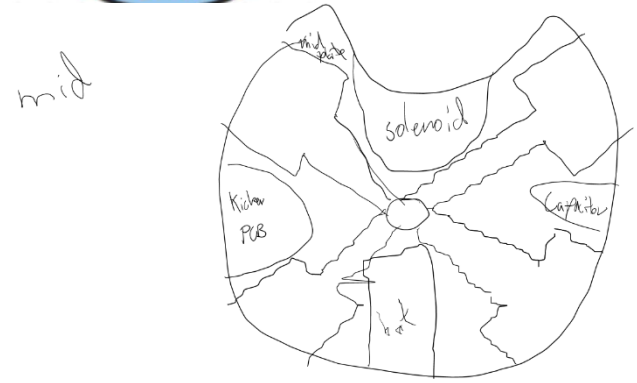
13. Rough Brainstorming of Design





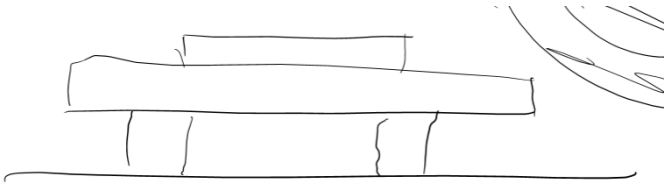
Base Plate: (BP)

- LSB idea:
  - Inner Circle (x24)
  - Outer Sections (x6*4 = 24)
  - X48 Total
  - (Outer detection
- Alter Capture guard for new size
- Will have to raise BP slightly to fit LSB guard without scraping



Mid Plate:

- Double Wheels
- (motor brackets moved out = battery not on angle = lower tssp ring = lower robot = lower COG)
- Wheels in corners
- 4 general frame parts (1 in each side)
  - Solenoid (front)
  - Kicker PCB (L/R)
  - Cap (L/R) (switches above cap)



- Battery (back)

Upper / Mid:
(similar to current, slightly lower tssp ring)
(upper drawing = main pcb, middle drawing = tssp ring, lower drawing = standoffs)
(add a spot for comm module on main)

14. Rough Notes for Improvements and Focuses

Category	Ideas / Options / Notes (Brainstorm + Hierarchy)
Overall Focus	• Priority = **Reliability + Durability**, not complexity. • Most hardware already works — aim for **consistency** and **fewer random failures**. • Reinforce weak points (PCBs, wiring, connectors, frame joints).
Compass & Orientation	**Issue:** Disconnecting / failing compass. **Possible causes:** Faulty module, loose attachment, PCB fault, other components interfering. **Ideas:** – Investigate if issue is compass itself or PCB connection. – Try **thicker PCB** and **thicker traces** on power/signal lines. – Re-solder all main components (poor joints may cascade failures). – Check **header quality** (looseness / fragility). – Revisit **fuse reliability** and **5V regulator** specs. **Compass-specific options:** – Replace with **BNO085** (more stable than current). – Reinforce connection: print small bracket or support under compass. – Add soft glue/tape reinforcement for vibration damping. – Consider **multiple compasses (2–3)** for redundancy — can detect and recover from disconnections automatically. – Use multi-compass system to isolate source of failure (sensor vs PCB).
Main PCB / Electronics	• Focus on **durability and robust soldering**. • Consider thicker copper / board layer. • Re-check **fuse blowing issue** and regulator reliability. • Use more secure headers and connectors. • Possibly revise **trace widths** on power lines. • Review components (e.g. MOSFET, 5V & 3.3V regs, Teensy 4.1).
New Additions	**General:** Plan early which PCBs we'll design (main, IR, light ring, kicker). • **CNC mirror:** design hyperbolic shape (use formula + simulate FOV).

	• **Capture guard redesign:** decide if one or two versions (lightweight vs open ball). – Consider adding **foam / sponge lining** for better ball control and softer contact. • **Frame strength:** check which sections flex or failed — strengthen or reinforce if needed. • **Wheels / rollers:** – Check if all 3 rollers touch ground. – Possibly lower wheel mounts slightly (to avoid LSB scraping). – Recut rollers from silicon tubing if quality was poor. – Double wheels = probably not worth it (minimal benefit).
Kicker (New)	• Confirm addition for 2026. • Design **Kicker PCB** (solenoid + capacitor + XL6009 booster). • Select solenoid model (A/B/C). • Integrate with main Teensy and camera logic. • Add kicker mounting support and ball-release calibration. • Optional addition (useful for ball retention and positioning).
Dribbler (New)	• Design lightweight, compact dribbler; ensure reliable gearing and motor control. • Needs testing — may complicate power management. • Only worthwhile if coding + ball control system improves first.
Sensors & Detection	**Light gate:** – Add to capture guard to detect ball possession. – Connect to main PCB (decide model + wiring). **IR PCB:** – Teensy 4.0 with TSSP58038 sensors. – Add RGB LEDs for diagnostics (only if time). **Light sensor ring:** – 32x LEDs + phototransistors + multiplexers. – Choose shape (outer, inner, or both). **Other sensor ideas:** – Ball detection via IR intensity / reflection. – Skip ultrasonics (not necessary with improved camera & mirror).
Component Decisions (Summary)	• **Main PCB:** BNO055 or BNO085, Teensy 4.1, OpenMV H7+/RT, HC-05 Bluetooth, fuses, regulators. • **IR PCB:** Teensy 4.0, TSSP58038 sensors, RGB LEDs (optional). • **Light Sensor PCB:** LEDs, phototransistors, multiplexers. • **Kicker PCB:** Solenoid, 10 000 µF capacitor, XL6009 booster.
Scope Management	• Don't overcommit — focus on getting **few features perfect** rather than many half-working ones. • Realistic inclusions: **Kicker**, **CNC mirror**, **IR communication**, **capture sensor**. • Deprioritise: dribbler (unless time), ultrasonics, unnecessary complexity.

13. General Restrictions

Dimensions & Weight:

- The robot must fit within a cylinder of 220 mm diameter, measured upright with all parts extended.
- Height must not exceed 220 mm.
- Maximum weight for Lightweight league: 1400 g in 2025.
- The "ball-capturing zone" (how far the ball can enter the front of the robot) must not exceed 15 mm.

Electrical / Power / Communications:

- Robots must not use mains electricity (i.e., plug into mains while playing).
- Maximum allowed voltage (general) is up to 48 V DC or 25 V AC.
- For the Lightweight league: the nominal voltage limit is 12 V for non-kicker circuits; the kicker drive is allowed up to 48 V.
- Communication between robots during gameplay (if used) must be on the 2.4 GHz band, and output power must not exceed 100 mW.

Sensors / Interference / Materials:

- Components designed to emit IR (for example, IR LEDs/lasers, IR distance sensors, LiDAR) are not allowed.
- The robot must not produce visible light that may interfere with the opposing robot's vision system.
- Robots must not be coloured orange, yellow or blue (since these colours overlap with ball/goal/field colours) unless that colour is occluded or taped over in a neutral colour.

Construction & Mechanisms:

- All active mechanisms (e.g., dribbler gears) must be covered with hard plastic or metal to ensure protection and safety.

Kicker

- Kicker power is subject to measurement/inspection. For example, the ball must not leave the opposing goal's penalty area after bouncing off the goal when tested.

14. Stability

Define Problem

The robots must be made from a sturdy yet long-lasting material to be structurally stable and impact resistant.

What type of material should be used for the robot's frame?

Background Research

The following table compares different materials that have been considered and previously tested:

Key Factors	**Material**			
	3D Printed PLA	Aluminium 6061	Carbon Fibre	Polycarbonate
Strength	**Low–Moderate**	**High**	**Very High**	**Moderately High**
Weight	**Very Light** density ~1.25 g/cm^3	**Light** density ~2.7 g/cm^3	**Very Light** density ~1.6 g/cm^3	**Light** density ~1.2 g/cm^3
Manufacturability /Accessibility	**Very Easy** 3D printable, low setup, accessible to anyone	**Moderate** requires CNC, machining tools	**Difficult** requires safe machining	**Moderate** requires laser cutting
Cost	**Very Low** Filament is cheap	**Moderate,** Affordable, and common	**Very High** expensive material and process	**Moderate High** More costly than PLA, cheaper than carbon fibre
Conductivity	**Not conductive**	**High** A thermal and electrical conductor	**Moderately High** Dependent on its structure, composition, and how it's manufactured	**Not conductive**
Durability	**Low** Degrades over long periods of time and over many impacts, non-heat resistant.	**High** Long-lasting, fatigue-resistant	**Very High** Flimsy in one direction, fatigue-resistant	**Moderate** Cracks after multiple impacts

Requirements

The material chosen must be:

- Relatively light weight
- Cost effective
- High Impact resistance
- Moderately easy to manufacture
- High strength

Plan for Solution

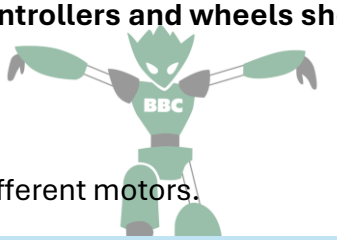
Our robot's plan is to use 3m thick Aluminium 6061 for its strength and durability while remaining relatively lightweight.

15. Movement System

Define Problem

For the robots to move smoothly, efficiently and effectively around the field whist outperforming the opponents in speed, power and control they must have a reliable, strong and agile movement system.

What type of motors/motor controllers and wheels should be used and how should they be positioned?



Background Research

The following table compares different motors.

	Type of Motor			
Key Factors	Pololu	Maxon DCX19 9V with 16:1 gear box	EC 45 flat Ø42.2 mm, brushless, 30 W	EC 45 flat Ø45 mm, brushless, 30 W
Recommended Voltage	12V	9V	12V	24V
Nominal Torque		11.3 mNm	57mNm	55mNm
No load Speed (RPM)		300 rpm	4390 rpm	4360 rpm
Dimensions		15mm in diameter	42.2mm in diameter	45mm in diameter
Weight		45.8g	75g	75g
Motor Controller used	L298N	VNH7070	ESCON 50/5	ESCON 50/5

	Type of Wheel	
Key factors	Single Omni-Directional Wheel	Double Omni-Directional Wheel
Load Capacity	Lower because rollers handle more stress	Higher because the load is distributed across more

	individually, reducing overall load capacity.	rollers, allowing greater weight support
Traction & Stability	Offers less traction, which can lead to slipping on smooth surfaces.	Provides greater traction and smoother, more controlled movement.
Size	Narrow and compact, suitable for smaller robots or limited spaces.	Wider area, offering better balance and surface contact.
Durability	Rollers wear out faster due to concentrated stress.	Increased durability because the rollers share the load and wear more evenly.
Maintenance	More affordable but may require frequent roller replacement.	Higher initial cost but lower maintenance frequency over time.

Requirements

➢

Plan for Solution

16. Ball Capture/Release System

Define Problem

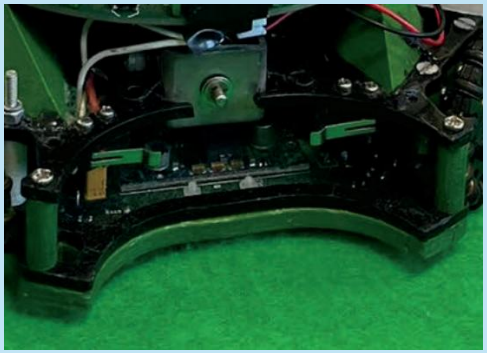
Our robots must have a reliable and controlled method for guiding the ball so that they can effectively capture and accurately release it to score goals from different ranges.

What shape should the capture zone be?

Background Research

The following table compares different shapes of capture zones.

Shapes of the Capture Zone

Key Factors	Curved	Rectangle	Rounded
			
Space Required	Matches the curve of the ball's edge, minimising wasted space.	Requires a large amount of space due to flat edges.	Matches the ball's edge, using space efficiently
Control & Alignment	Provides good ball control thanks to the curved shape, which helps guide and align the ball toward the centre of the capture zone.	Offers limited control because of the large flat area, making alignment more difficult.	Delivers excellent control within the capture zone due to increased surface contact and the rounded curve, allowing smooth alignment to the centre.
Ball Entry Accessibility	Allows a large amount of space for ball entry due to its open shape.	Provides a large amount of space for ball entry because of its wide design.	Offers less room for ball entry, requiring more accurate positioning for successful capture.

Requirements

- Can control the direction of the ball relative to the robot when moving forward
- Aligns the ball in the centre of the capture zone effectively
- Minimises the space required to implement in the design

Plan for Solution

16.1 Kicker System

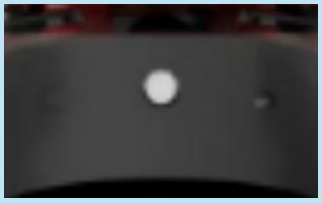
Define Problem

For our robots to effectively and accurately score goals from a distance, they must have a reliable and powerful kicker system to score goals flawlessly.

What type of shape should the kicker be?

Background Research

The following table compares different shapes of solenoid attachment.

	Shape of Attachment	
	Bar solenoid attachment	Dot solenoid attachment
Key factors		
Power Transfer To a point	Distributes power along the bar, resulting in a weaker kicking force.	Concentrates power at a single point, producing a stronger and more direct kick.
Accuracy and control	Less precise but allows for a wider contact area, providing more chances to make contact with the ball.	Highly accurate when the ball is fully positioned in the capture zone, allowing excellent control during the kick.
Space required	Requires more physical space due to the length of the bar.	Compact design requires less space, ideal for tighter builds
Reliability	Consistently delivers a kick even if alignment is imperfect, though accuracy is reduced.	Reliable only when the ball is correctly positioned, but it offers higher accuracy and power.
Kicking Consistency	Provides a steady but moderate kick force across different positions.	Produces stronger but less consistent results if the ball is misaligned.

Requirements

- Powerful and consistent kick
- Accurate kick
- Minimises the space required for implementation

Plan for Solution

We plan to use the dot-shaped kicker design as it allows for a more powerful kick compared to the bar design. Additionally, it minimises to space needed to implement, which decreases the weight and increases extra space in the robot

16.2 Dribbler System

Define Problem

For our robots to effectively and efficiently take possession of the ball whilst on the field, they must have a reliable way to keep possession of the ball whilst moving.

What type of motor should be used, and what would be the best method to keep possession of the ball?

Background Research

The following table compares different shapes of dribblers.

	Shape of Dribbler	
	Line-shaped Dribbler	Dual Cone shape dribbler
Key factors		
Ball control *Stability when moving* *Positioning* *Moldability*		

Requirements

-

Plan for Solution

17. Main Processing/Communication System

Define Problem

For our robots to effectively process code and control all aspects of the mechanics, they must have a main microcontroller, a compass, and a communication module that is reliable and effective during games.

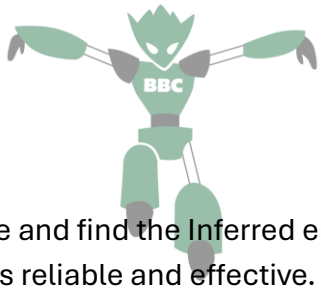
What type of main microcontroller, compass and communication module should be used? How many of each and why?

Background Research

Requirements

➢

Plan for Solution



17.1 Ball Detection System

Define Problem

For our robots to precisely locate and find the Inferred emitting ball, they must have an effective detection system that is reliable and effective.

What type and what shape should the Inferred sensing board be, and how many sensors should be used?

Background Research

	Shape of Board

Requirements

➢

Plan for Solution

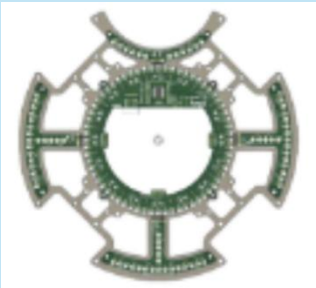
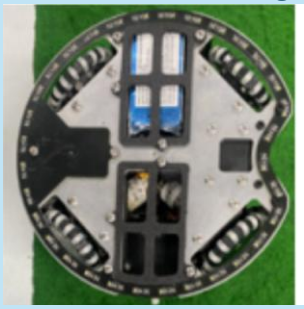
18. Line Avoidance System

Define Problem

For our robots to effectively, precisely, and efficiently avoid the white line, they must have a light detection system that is reliable and durable while on the field.

What shape should the light sensor board be, and how many LEDs should be included?

Background Research

Key Factors	Duel Outside sections and Inside wheel ring	**Shape of Board** Outside Wheel Ring	Inside wheel Ring
			

Requirements

➢

Plan for Solution

19. Vision System

For our robots to precisely and effectively view both the coloured goals and orange ball they must have a camera and mirror system that is reliable and durable.

What type of camera should be used? What shape of mirror should be used and how should it be positioned and adjusted?

Define Problem

Background Research

	Shape of Mirror Mount

Requirements

➢

Plan for Solution

20. Summary of Decisions

Frame

- 220mm diameter limit therefore our robot will be 210mm
- Aluminium 6061 for endurance due to 3D printed PLA being flimsy and brittle

Movement

- 4x DCX 19 Maxon Motors in X shape
- Double omni wheels for extra grip at high speeds with thin guard around them for protection

Vision System

- Acrylic tube for a full 360 view of the field with compact yet accessible adjustment plates to Centre mirror
- Camera and hyperbolic mirror for goal tracking and orange ball detection
- Hyperbolic mirror (track ball and goals, potentially also for coordinate system)

Ball Capture

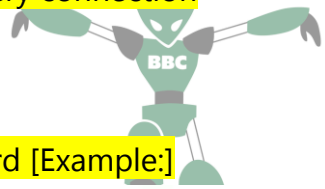
- Curved capture zone for more contact area
- https://junior.robocup.org/rcj-ir-soccer-ball-2026-changes-announced/ (42mm diameter ball, use ir and camera to track) ('ball-capturing zone of up to 1.5 cm')
- Spherical cut out, with space for kicker)

Main Board

- 24x TSSP58038's for ball detection on 1x Teensy 4.0 for more accurate strength and direction readings
- 24 RGB LED's matching TSSP's (don't have to use them, but it's for the funnies, and we can use it to showcase the ball direction, also can be used to detect

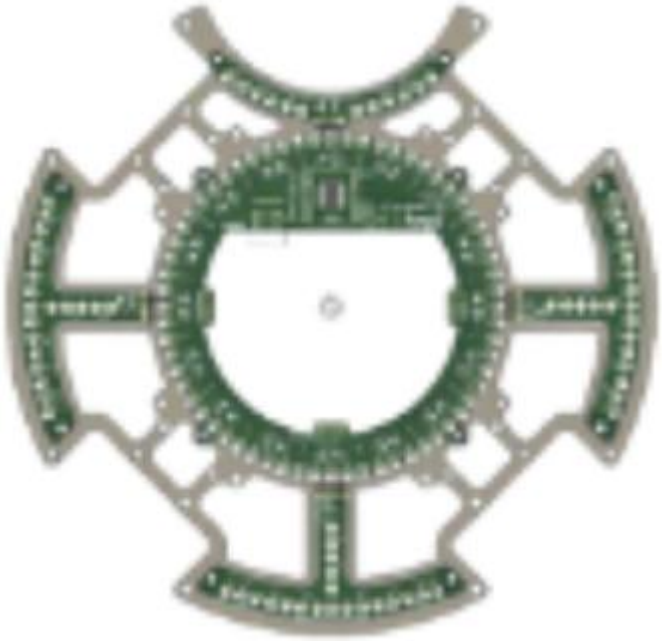
the different roles of the robots to see if Bluetooth is connected and when it switches) (PLEASE LET US DO THIS)

- HC-05 Bluetooth modules for switching strategy
- BNO005 for compass correction
- OpenMVH7-plus for orange ball detection, goal tracking and localisation
- DCX19 Maxon motor for dribbler motor
- 4x VNH7070's motor controllers
- Teensy 4.1 as main micro controller
- 5V and 3.3V regulators for logic line
- Voltage divider to read battery voltage
- 1x switch for discharging capacitors
- 1x switch for robot power on/off
- 1x switch for logic power on/off
- 1x switch for RGB on/off (if we have it) (AGAIN PLEASE LET US ADD RGB)
- 1x switch for dribbler enable/disable
- XT60 extension for battery connection
- FFC down to LSB



Light Sensor Board

- 2 rings light sensor board [Example:]



(either 32, 48 or 64 LED's and no straight parts aiming towards the middle)
For fast line detection, space with wheels going to ward the edge of the robot so there is room for the solenoid, and inner ring for defender.

Kicker Board

- 1x 10000uf capacitor connecter for storing voltage for solenoid
- 1x Solenoid connecter for kicking the ball
- Voltage divider for capacitor reading.

Wheels didn't have enough grip when robots ran at high speeds when surging; therefore, they slid on the field.	**A double-layered wheel should be used for more grip on the field due to its increased surface contact area with the ground.**
Battery position and replicability were inefficient and not safe	The battery connector extension should extend out more, and the battery position should be (just improve battery connection position and cable length)
The top of the Main Board was inaccessible without taking off the adjustable tube/mirror mount	The size of the adjustable tube/mirror mount should be decreased to not cover the entire main board.
The bottom of the Main board was inaccessible without taking apart the robot	Adjust how the TSSP ring is mounted to the robot. Instead of having the TSSP ring standoffs screw into the mid plate and the motor holder
The light sensor board scraped against the painted white lines at States	Adjust the position of the light sensor board so it's not close to the ground
The light sensor board could have shorted when positioned against the aluminium base plate	Design a divider between the Light sensor board.
The international communication module was not considered in the design before ordering/printing	Add a section for it with it, for easy use, as well as testing with the module beforehand

Planning References

1. https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/

2. https://forum.arduino.cc/t/hc-05-bluetooth-module-interference-with-heart-beat-sensor/637639

3. https://forum.arduino.cc/t/sending-a-value-between-arduinos-using-hc-05-bluetooth-modules/859376

4. https://www.electronicwings.com/sensors-modules/bluetooth-module-hc-05-

5. https://github.com/SamGargRobotics/Hyperion-BBC-Robotics/blob/main/2025%20(RCJ%20Lightweight%20Soccer)/Software/lib/Camera/Camera.cpp.