FG&B TEAM DESCRIPTION PAPER

Robocup Junior Soccer – Open Division
Brisbane Boys' College, Queensland, Australia

Team Introduction

Team FG&B is a team of final year high school students from Brisbane Boys' College, located in Brisbane, Queensland, Australia. The team have been working together for the past three years competing in the Robocup Junior Australia competition in the Open Soccer Division. This division is very similar to the international open competition, except Australia is yet to adopt the passive ball. Over the past three years, we have achieved the following results:

Queensland State Championships 2015	2nd Place
Australia National Championships 2015	3rd Place
Brisbane Regional Championships 2016	1st Place
Queensland State Championships 2016	2nd Place
Australia National Championships 2016	5th Place
Brisbane Regional Championships 2017	1st Place (Undefeated)
Queensland State Championships 2017	1st Place (Undefeated)
Australia Nationals Championships 2017	1st Place (Undefeated)

The 2017 National Championship was held on the 23rd and 24th of September in Brisbane, Australia and we are very happy with our results. We were hoping to qualify for the international competition this year in Nagoya, Japan, but unfortunately missed out due to some robot troubles in the finals at the national competition last year. This year we have come back significantly stronger than in previous years, and our results are showing this as we are yet to lose a match in the 2017 competition. We believe we are now ready to compete on the international stage and challenge ourselves against some of the best in the world in our competition.

Figure 1 - The Team (William Plummer, Alistair English, Thomas Fraser, Cooper Richmond, Thomas Hulbert)

Hardware

The 2017 robots have been designed around a modular design that allows the robots to be easily disassembled to allow quick repair, and for other aspects to continue to be worked on whilst upgrades are being performed. This setup can be seen below:

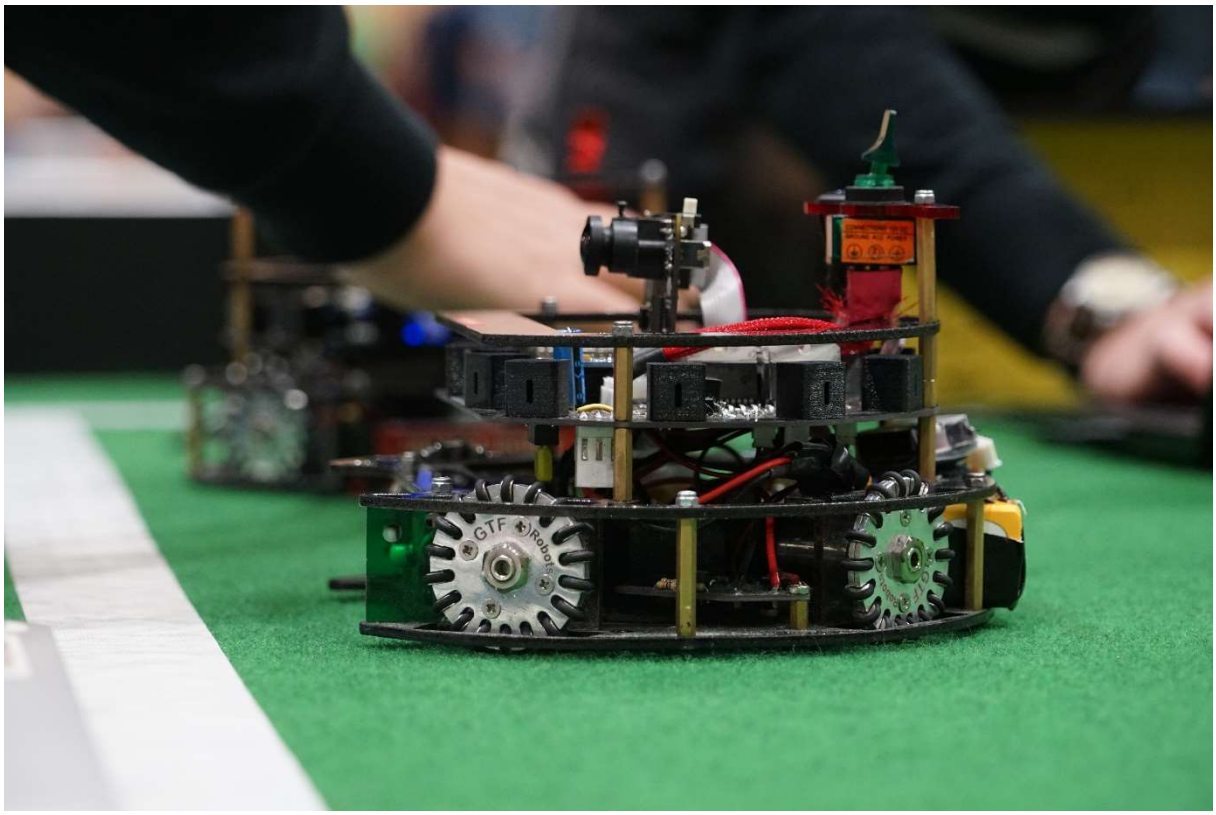

Figure 2 - Side On Image of 2017 Robot

The lower section containing the motors, solenoid, solenoid circuit and battery can be removed from the upper section containing the main PCB with only 4 screws. This means that access to the critical parts of the robot, such as the PCBs can be achieved quickly and efficiently. This setup has so far been effective in minimising time wasted when repairs are carried out.

The lower portion of the robot features four Maxon DCX19 brushed motors arranged in an omnidirectional arrangement to allow holonomic control of the robot. This allows the robot to move and face any direction, at the same time. The robot also features a Takaha Kiko solenoid to allow kicking of the ball towards the opposition goal. This solenoid is controlled using our self-designed solenoid circuit, which allows charging up to 70V and adjustment to ensure the robot always kicks at the maximum, legal strength. The introduction of a kicker to our robot has made a significant difference to our gameplay, as we now do not have to rely on beating the opposition to their goal. A “light gate” on the front of the robot is used to detect whether the robot has successfully got the ball in the capture zone, and triggers the solenoid to kick.

The lower section of the robot also holds the 1300mAh 4s LiPo battery. These batteries easily last a half, meaning that the robot is always staying at top speed. The robot also has battery monitors to prevent accidental over discharge of our LiPo batteries. On the bottom of the robot, there is a light sensor circle to allow the robot to detect and track the line across the robot, meaning that the robot can pass fully over the line, and still return to the field. There are 19 light sensors arranged every 18 degrees around the robot, with one missing at the front due to the solenoid.

The upper portion of the robot is home to the main PCB and most of the robot's sensors. These include a pixy cam to track the goal, an IMU to track the robot's orientation and the infrared sensors used to track the IR ball used in the Australian competition. The data from all our sensors is processed using three Teensy 3.2 microcontrollers, as seen in the diagram below:

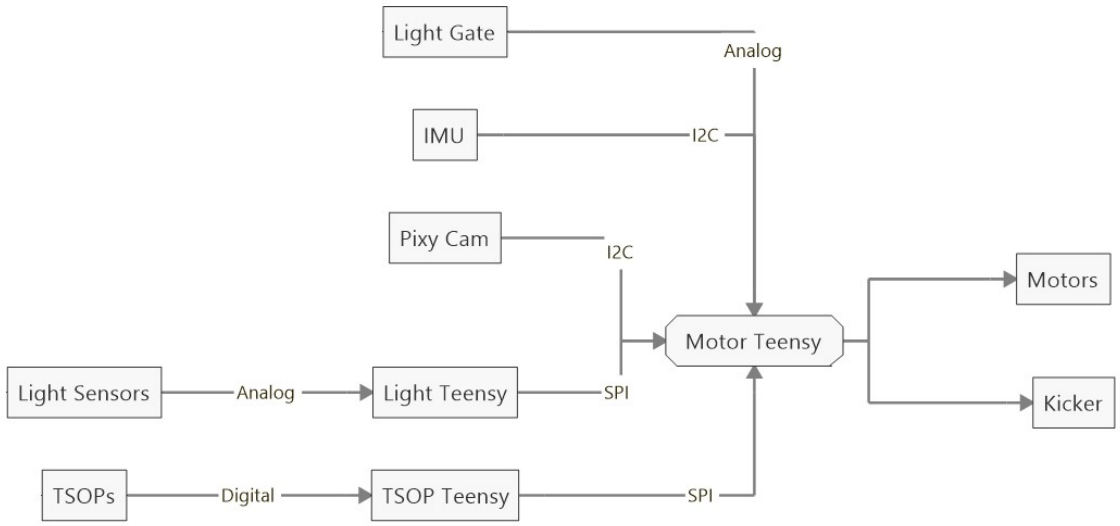

Figure 3 - Main PCB Flow Chart

As the Open Competition at Robocup Asia Pacific will use a passive ball as per the international rules, the robot will be modified accordingly to suit the competition. A 360-degree camera system has been designed making use of an Open MV M7 as the vision sensor and a polished stainless-steel hemisphere as the mirror. A render of the designed system can be seen below:

Figure 4 - 360 Degree Camera system for FG&B's current robots

This system can be easily mounted to the top of our current robots meaning that no major redesigns are required and a large percentage of our current code and tactics can remain as they are currently. The camera system will simply replace the TSOPs and the TSOP Teensy thus reducing the robot down to two microcontrollers and the camera system. The x and y coordinates of the ball in the cameras view can be used to determine the angle to the ball relative to the robot using the atan2 function and the approximate distance to the ball can be found using Pythagoras theorem. This means we simply replace our current ball tracking method with this system and the rest of the robot can remain as is as seen below:

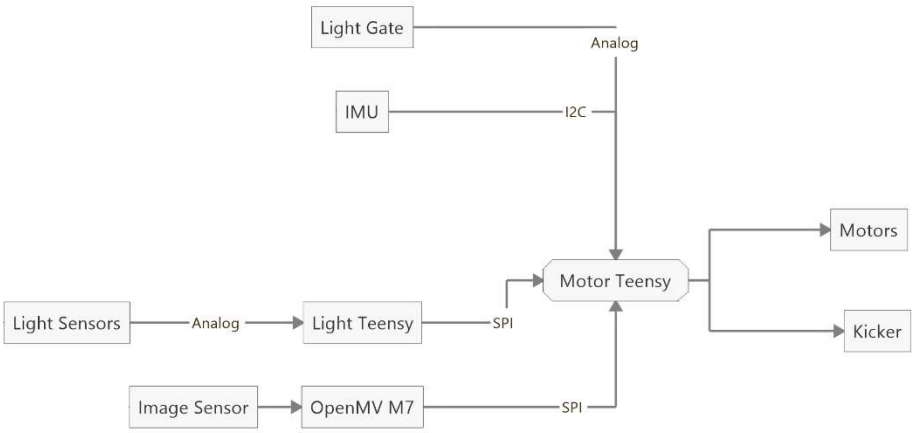

Figure 5 - Hardware Flowchart with 360 Degree Camera System

Software

The software currently being used on the 2017 robots is the result of 3 years of iterative development and testing, both independently and in competition. The code is split across three Teensy 3.2 processors and functions as shown below:

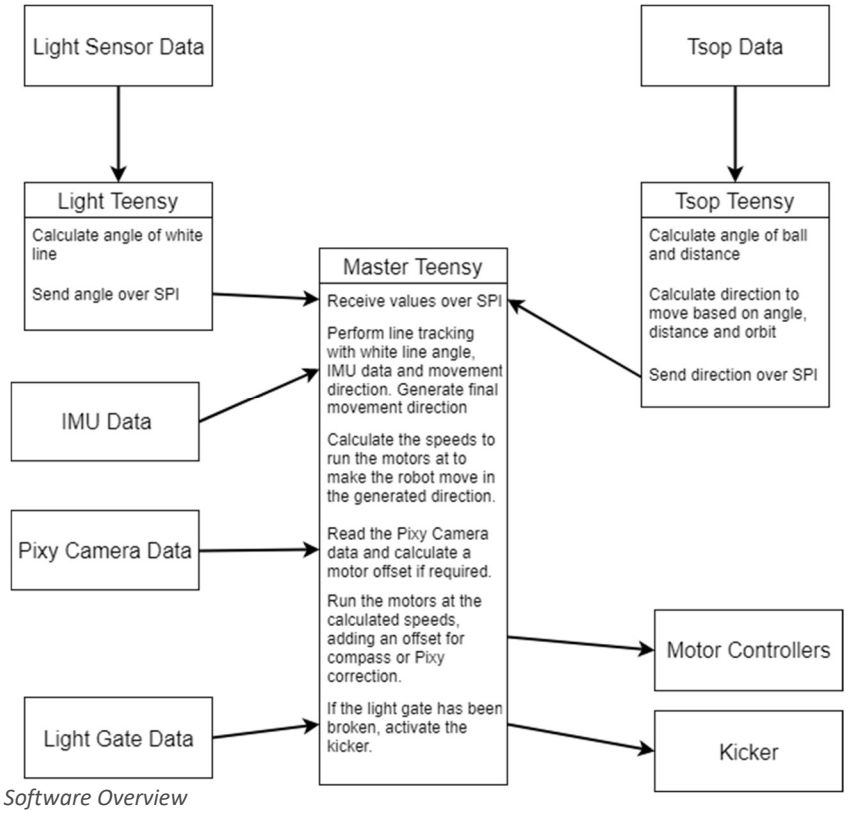

Figure 6 - General Software Overview

The software employs efficient methods of programming such as the use of class-orientated programming and external libraries. Libraries are used to declutter the software and make it more readable.

TSOP Teensy

Ball Locating

The current setup for the Australian competition makes use of TSOP31140 IR sensors to track the ball. These sensors then return a high signal if they are receiving infrared light. The sensors are read repeatedly 256 times so that the sensor that is seeing the ball the strongest can be determined. This is then returned as an angle from the front of the robot indicating the direction of the ball.

For the Robocup Asia Pacific Competition, this teensy will be replace with an OpenMV 7 camera with a 360-degree viewing system. This system gives a view similar to below:

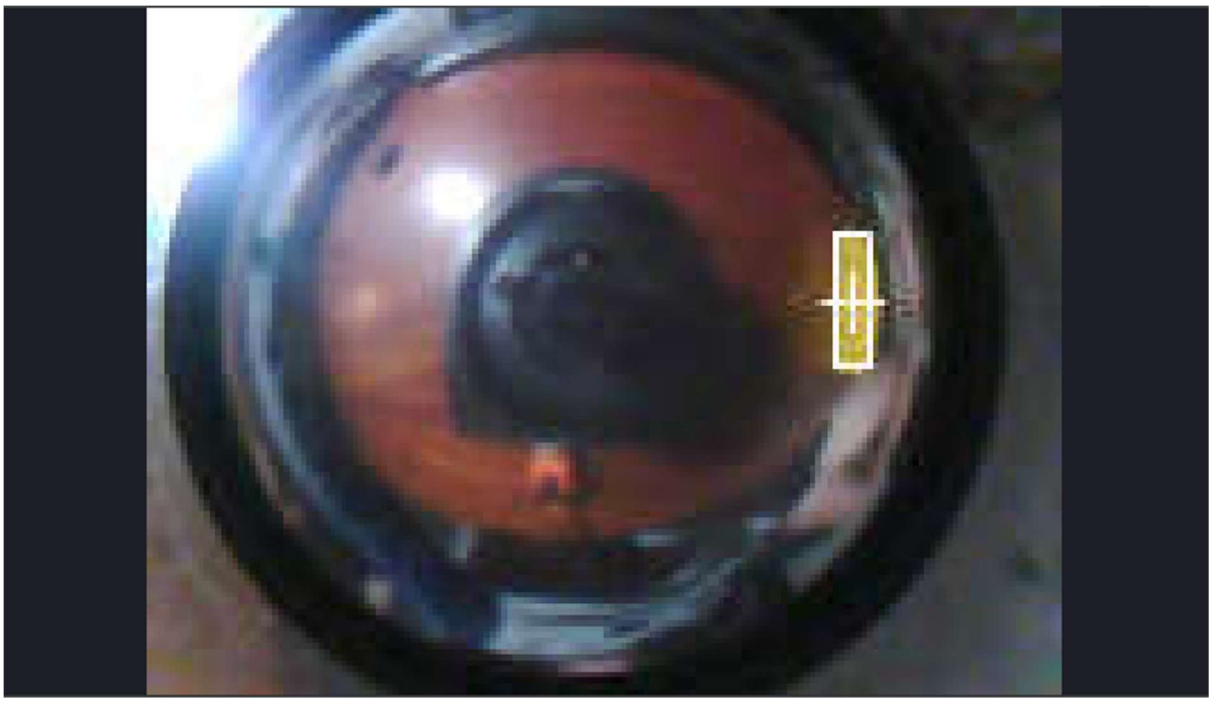

Figure 7 - Our 360 Camera tracking a yellow rectangle

The ball angle is found using the atan2 function, which converts the (x,y) coordinates of the object to an angle relative to the origin (the centre of the image). The (x,y) coordinates can also be converted to a ball distance value using Pythagoras theorem, allowing the robot to move directly to the ball when far away and then orbit when close, using the following method.

Orbit

As the current robot always faces forward, the robot must "orbit" around the ball so that it is behind it and thus can push / kick it into the goal. To determine the angle to move, the ball angle is passed through a function that generates the movement angle. If the ball angle is greater than 90 degrees and less than 270 degrees, the robot orbits at 60 degrees from the ball angle backwards. This means that for example if the ball angle was 130 degrees, the robot would move at 190 degrees. If the ball angle is in the front half (less than 90 degrees or greater than 270), the angle is passed into the following function:

$$Orbit\ Angle = Ball\ Angle \pm \frac{Ball\ Angle}{120} \times 90$$

$$* The\ \pm is\ positive\ if\ ball\ angle < 90\ degrees\ and\ negative\ if\ ball\ angle > 270\ degrees$$

This function returns the orbit angle for the robot to move along to correctly position behind the ball. This angle is then sent using SPI to the master teensy.

Light Teensy

White Line Angle Calculation

To calculate the direction to move to stay on the field, the code considers groups of sensors (named "clusters"). In an iteration of the main loop, the program loops over all light sensors and identifies clusters of adjacent sensors that are seeing the line. As the robots physical light sensor array is arranged in a circle, the light sensors are assigned x and y coordinates on a unit circle according to their degree of rotation around the circle. For each cluster identified, the Cartesian coordinates of the sensors in the cluster are averaged to obtain a centre point for the cluster. Then the centres of each cluster found are averaged to form a point that would exist on the white line. From there a line is drawn from the centre of the robot to the point and the angle that this line points is the angle of the line to the robot. 180 degrees from this angle is the direction the robot should move to avoid the line and this is the angle sent over SPI from this Teensy.

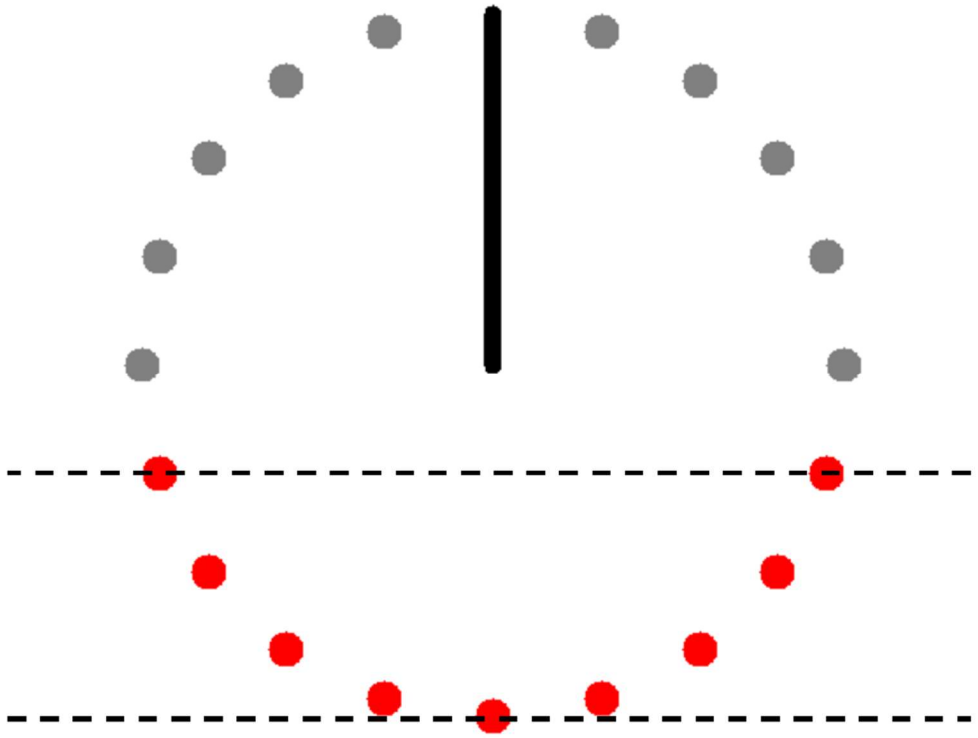

Figure 7 - Diagram showing the line (dashed), the sensors seeing the line (red) and the resulting angle

Master Teensy

Line Tracking and Movement Algorithm

To determine the direction of the robot the Master Teensy takes into account the data from the Light Teensy, the TSOP Teensy and the IMU. When the program first receives all data, the first thing it performs is line tracking. To do this, the code first adjusts the raw line angle from the Light Teensy with the IMU heading of the robot to create an "absolute" heading. This heading is now relative to the field rather than the robot. The heading is then processed following the flow diagram below:

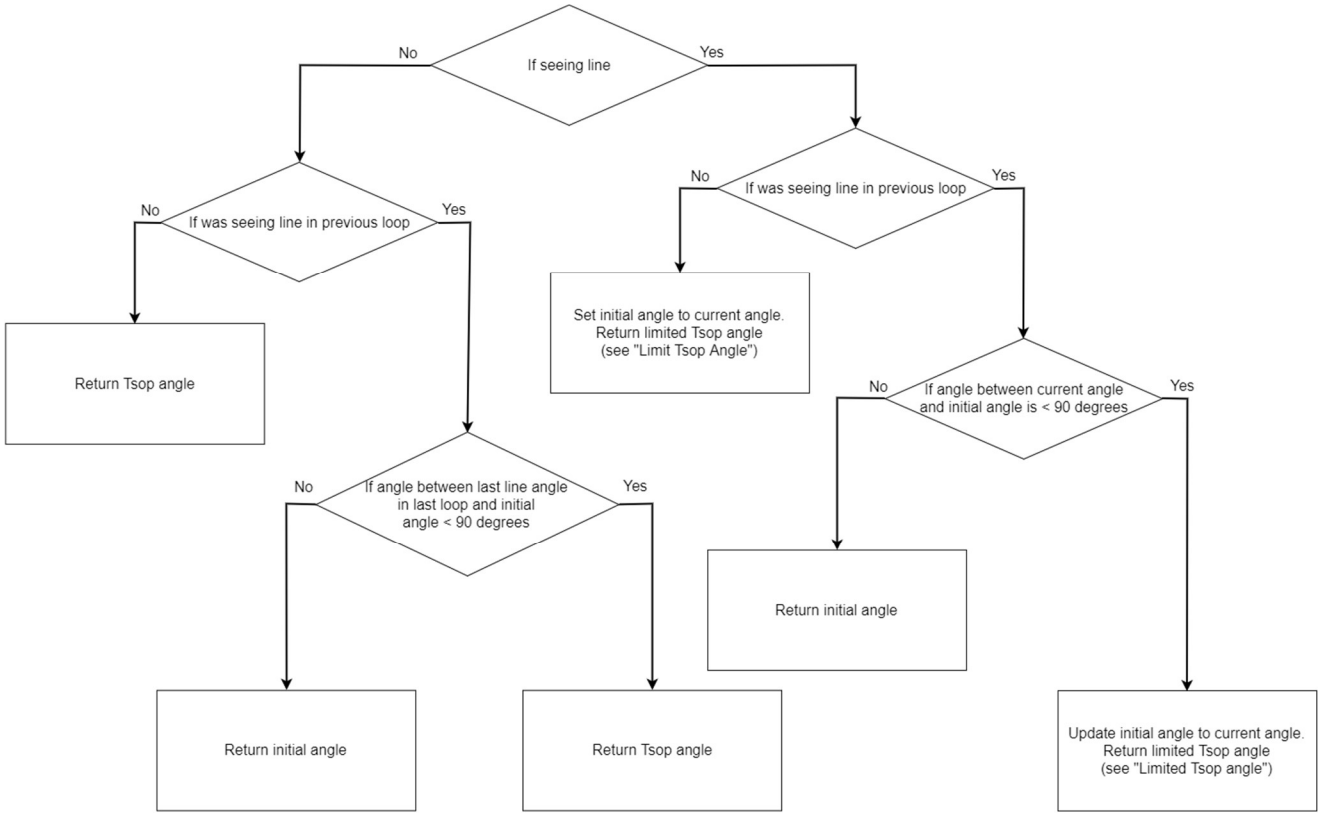

Figure 8 - Line Tracking Logic

Limited TSOP Angle Using Light

When the robot has recognised that it is out, its movement is also based off the orbit direction. The robot checks to see if the angle between the orbit direction and the light angle is less than 45 degrees. If it is, the robot moves on the orbit angle allowing it to continue to chase the ball. If the difference is greater than 45 degrees, the robot moves on the "movement boundary" closest to the orbit direction.

$$Movement\ Boundaries = Light\ Angle\ \pm 45$$

This means that the robot can continue to move towards the ball, whilst also returning to the field. This means that whilst the robot is orbiting around the ball, it doesn't bounce unnecessarily reducing the time the robot takes to orbit the ball.

Motor Speed Calculations and Compass Correction Offset

Motor speed calculations are one of the final steps performed in the loop of the program. The motor speed calculations consider all factors including line tracking, ball direction, compass correction and goal tracking. All these factors are correlated into a single PWM value (ranging from -255 to 255) that is written to each motor. The motor speed calculations first determine the optimal angle to move on whether that angle be to avoid the line or track the ball. This angle is then used in an equation to find the speed value to set to each motor. The calculation also considers the angle of the motor relative to the centre front of the robot. This calculation uses a cosine function.

$$Decimal\ Percentage\ Speed = \cos(motorAngle + 90 - angle)$$

To add rotation to the robot, a set rotation is added to all motor values allowing the robot to still orbit and move correctly while rotating. As our motors are unable to go above 100% speed or a PWM value of 255, the direction values must be correctly scaled to ensure that the robot will always rotate correctly. The following pseudo-code shows how this is completed:

```
motorValues = an array containing 4 Decimal Percentage Speeds for the 4 motors
howFast = the final speed as a 0-100 value that we want the robot to move at
    array[4] scaleMotorValues(motorValues[4], rotation)
    speedScaled = 255/largest(motorValues)
    motorValues[] = motorValues[] * speedScaled + rotation
    speedScaled2 = 255/largest(motorValues)
    return motorValues * speedScaled2 / 100 * howFast
```

The motor values for each motor are then set as the PWM input for each respective motor controller. The motor controller is also passed a direction and a brake value. The direction passed determines if the motor spins forwards or backwards, this is determined by the sign in front of the PWM speed given. The brake input will brake the motor.

Solenoid and Light Gate Operation

Determining when to kick the ball is controlled by the light gate. The light gate which consists of a laser and LDR, is 'broken' when the ball is within the capture zone of the robot. The master teensy is informed of this by reading an analog pin. When the reading from the pin falls below a certain threshold, the robot knows that the ball is in the capture zone and thus can be kicked. The robot preforms a check to make sure that the solenoid circuit has had enough time to recharge before kicking. The solenoid is controlled by the master teensy by a single digital pin. To perform a kick, a digital pin must be set to low of a period of 1 millisecond before returning to high. This gives the capacitor enough time to full discharge. A period of 2 seconds is given between kicks for the circuit to recharge.

Thank you for your time and consideration for us into the Asia-Pacific Robocup Competition and we look forward to your response.

Regards, FG&B