Sheffield Autonomous Racing Car Final Report

Andrew Huff, David Chung, Hamish Sams[1]

***Abstract*— Sheffield Autonomous Racing Car (ShARC) is a self driving car designed to compete in the NXP cup. This paper covers all parts of the project including motor selection, computer vision and control methodologies starting from a blank canvas. The project has implemented many technologies such as Generative Adversarial Neural Networks (GANs), Hough Transforms and Brushless Direct Current Motors (BLDC) successfully and unsuccessfully, both covered here. Here we show the design, implementation and partial testing of the entire ShARC. This project was ended unexpectedly and abruptly by the Coronavirus Disease 2019 (COVID-19) leaving many tasks infeasible to complete. The project ended in a partially finished state with many features designed but not implemented due to the restrictions.**

***Index Terms*—Brushless DC Motor, Computer Vision, Image Processing, Neural Networks, Self-Driving Car, Servo motor, Wired Communication**

I. Introduction (David)

'Cars that drove themselves while drivers relaxed' – the grand proposal set by General Motors' Futurama exhibit in 1939 in that year's World Fair detailed some of the first instances of the concept of Advanced Vehicle Control Systems (AVCS), vehicles capable of autonomous control [1] . Since the inception of this concept, the idea of self-driving cars has been of continuous interest for commercial and academic bodies, in progression spanning from the 1950s to the present day. Notable developments are the early investigations by General Motors during the 1950s [1], the 'Intelligent Vehicle' [2] and the Personal Vehicle System from Japan in the 1970s and early 1980s [3], Autonomous Land Vehicle [4] and NavLab [5] projects in the United States across the 1980s, and the European PROMETHEUS project in the 1990s, which most notably achieved a 4586 km journey from Munich, Germany to Odense, Denmark whilst achieving 95% of travel done with automated steering [6].

Today, there is a greater engagement in AVCS than ever before. Examples include the release of modern commercial vehicles by pioneering companies such as Tesla and Amazon with effective autonomous modes [7]. There has also been a rise of well-funded, distinguished public competitions dedicated to autonomous vehicles such as the DARPA grand challenge. In these, teams apply internationally to generate innovative solutions to urban, off-road and subterranean scenarios.

Lesser annual events such as the Race On USC Self-Driving Car Competition and the NXP cup also provide opportunities for smaller teams to develop autonomous solutions to racing cars, with complex navigation and decision making isolated to smaller tasks. In particular, the Europe Middle-East and Africa (EMEA) NXP Cup challenges teams to develop an autonomous $1/16^{th}$ scale car to compete in a time trial of an intricate track, as well as solve additional challenges. These are based on isolated tasks that road viable autonomous vehicles would handle, such as sign recognition and object detection. Points are given for the ranking in the time trial as well as for each successful challenge completed. For UK teams, the NXP Cup consists of two stages: A national qualifier between six teams held in Edinburgh and the EMEA finals in Bucharest. Winners of the Cup are awarded €3000, with an additional €1000 being awarded for the winner of the secondary Electromaker Innovation Prize.

This paper details the first year of development of an autonomous $1/16^{th}$ scale car, with the intent of establishing a persistent project that teams can iterate on and compete with in future NXP cups. Over this year, the scale car was completed with a speed controller, and progress was made on computer vision steering control using a Pixy2 camera as well as the ranging systems for the emergency brake and obstacle avoidance challenges. Due to COVID-19, the project was terminated early in March to comply with social distancing measures, but as the NXP Cup has been postponed from May to late 2020, the opportunity to compete in the 2020/2021 competition is still possible.

II. Aims and Objectives (Andrew)

The project aim was to design and build an autonomous $1/16^{th}$ scale racing car to compete in the EMEA NXP Cup competition. All five challenges which make up the competition were to be competed in.

The aim of the main race was to complete a single timed lap of a circuit in the shortest time. Track pieces were formed from two black lines on a white background and could only take on five standardised shapes: a straight, a 90˚ bend, a 180˚ bend, an intersection and a chicane. To complete a lap the car had to maintain at least two wheels within the track at all times. The full rules and track shapes could be found in [8] and [9].

The additional challenges included: the figure-of-eight, where the winning team would complete the most number of laps of the track in one minute; obstacle avoidance, where the

[1]This 4^{th} year Group Project was performed at the Department of Electronic & Electrical Engineering at the University of Sheffield, UK It was supervised by Luke Seed (e-mail: n.seed@sheffield.ac.uk) and Jiabin Wang (j.b.wang@sheffield.ac.uk) .

car was to avoid a 200 mm white cube placed at random on the track; speed limit detection, where the car was to significantly alter its speed in response to visual speed marker 'barcodes'; and emergency braking, where the car was to come to a halt without hitting a black cuboid spanning the width of the track placed at random.

The objectives set in the project initialisation document were revised in the interim report. Secondary objectives such torque vectoring were dropped due to time constraints against the marginal improvements which such systems could provide. Other objectives were changed to use more developed products in the interests of saving time. For example, the full development of an OpenCV based computer vision system was changed to use the Pixy2 camera and microprocessor board, which contained built-in computer vision algorithms. The project objectives were:

A. Develop a car

A scale car which complied with competition rules was required to compete. In order to rank well in the competition, the car needed to be performance orientated so that the navigation control systems were not limited by a sub-standard motor and chassis. The car needed to have a chassis which would allow simple mounting of additional control electronics.

B. Computer vision steering control

A computer vision based navigation system was to be developed in conjunction with a closed loop controller to autonomously steer the car. The competition rules stated that a camera must have been the primary navigation sensor. A commercial computer vision board such as the Pixy2 camera could be used. The system needed to be able to deal with intersections as well as navigate around all track pieces. As a secondary objective, such a system should be capable of taking a 'racing line' to improve lap times.

C. Object avoidance

A system was to be developed which would be able to detect the position of a 200 mm white cube against a white background relative to the car. The system must also be capable of altering the route of the car to drive around the obstacle without leaving the track.

D. Emergency braking

A system was to be developed which could detect the presence of a black 200 mm tall cuboid spanning the width of the track. The system must then cause the car to stop before hitting the object.

E. Speed limit awareness

A system was required to detect two known speed limit barcodes placed on the track and then adjust the car's speed up or down in response.

III. Economic, legal, social, ethical and environmental context (Hamish)

The self-driving car ethical debate is an extended one given the mix of autonomous and non-autonomous cars each with lives at stake. This model racing car instead does not post any damage to life except in the case of an extreme engineering failure in which risk assessments have been completed and followed over the course of the project. The methods used in this self-driving car are unlike all else due to the lab/test environment of driving, aka in an enclosed black and white track and thus offers little information to life at risk systems being developed. Smaller autonomous vehicles, like those used for deliveries, are more similar due to the lack of endangerment but can lead to less jobs as they are replaced by humans, in this pre-pubescent state these machines are unlikely to start costing jobs but are instead creating them for researchers and technicians and simply moving the jobs. In the future ethical decisions may need to be taken to protect jobs once the technology becomes advanced enough to be self-sustaining. Legally self-driving cars can be a gray area due to most laws being developed well before technology had the ability to replace human jobs but likely the repercussions will be either the user or the company depending on the cause of the issue or well hidden small print, thus creating a well designed safety system is of utmost priority to protect us as engineers and the users physically and legally.

A large legal and ethical question is of data security. Many systems such as GPS can and will track position but as more cars become technologically advanced, will this data be stored and does that data belong to the user, manufacturer or government. This is a huge question that will likely change depending on region of the world but for our case any information stored is purely local to the car and has no ability to be abused or be useful outside the contexts of this project.

IV. Literature review

A. Chassis and Drivetrain (Andrew)

The most relatable field of research to the project was line following robots. These typically have a simple chassis in a tricycle arrangement with two directly driven wheels and a castor wheel. They are referred to as differential drive mobile robots (DDMR). The motion of DDMRs have been well studied and modelled in line following applications [10]. They are also simple to control with effectively one actuator as the magnitude of the speed sent to each wheel is the same. Rotation is achieved by driving wheels speed with the same magnitude but in opposite directions while forwards motion is achieved by driving both wheels in the same direction. Rotation can be achieved without forwards motion which makes DDMRs highly manoeuvrable [11]. However, DDMRs are unstable at high speeds due to their tricycle arrangement and that any discrepancies between the motor speeds will cause the robot to rotate when moving forwards. High speed performance has been improved using Ackermann steering and a four wheel robot. However, this came at the expense of manoeuvrability [12].

Initial research looked into hobby remote-controlled racing car vendors as these markets were already well established. The available cars were close to the specifications required for the competition track and had already been designed with competition and performance as a requirement. Chassis were typically four wheel drive (4WD) in larger cars, with a mixture of two wheel drive (2WD) and 4WD in the mid-sized category. All racing level cars used a single motor and Ackermann style

steering; two motor differential drive was only seen at the budget end of the scale.

The majority of line follower robots in published research use brushed DC (BDC) motors driven by a H-bridge [10] [11] [12]. The reasoning is rarely stated, so it can only be presumed to be because of the simple control offered by BDC motors. However, it is important to note that none of these studies intended to develop a fast point to point vehicle. Once again looking at the hobby remote-controlled racing car market, there is a split between BDC motors used in budget focused cars and brushless DC (BLDC) motors in performance focused cars. This is due to the increased power to weight ratio offered by BLDC at the expense of increased motor and controller cost.

Due to the secretive nature of successful team designs entered into the competition, the biggest insights came from discussions with a winning contestant from a previous year. Although the competition differed in that the car was meant to follow a single black centre-line and could use a line scanner, the major concepts were still applicable. The design used Ackermann steering with independently driven rear wheels, similar to [12]. However, unlike [12] in which independent wheels were intended to prevent slippage during off road terrain which can result from mechanical differentials, the contestant used independent wheels to over-drive the outer wheel in corners to attempt to improve cornering by torque vectoring. BDC motors were used but only due to previous competition rules, it was advised that BLDC motors were used instead. Motor speed was controlled by an open loop H-Bridge controller. It was found that the speed of the car was more limited by the traction than it was by steering control, so any advancements such as 4WD would benefit lap times. Car speed was set proportional to the error between the track centre line and the car's trajectory. This caused the speed to decrease when approaching a corner as the car would begin to deviate from the centre line. However, the effectiveness of this method was questioned as the car would have no prior warning of a corner due to the line-scanner's position directly in front of the car, resulting in limited braking time.

B. Computer vision (Hamish)

The line following robot is a common project that is constantly improved by different methodologies and systems. A method commonly used is image segregation to chunk images into blocks then classifying the chunks to create a line matrix as seen in [13]. These methodologies are commonly based around a single black line compared to double lines as this is required to track, this adds complexity but the base concepts can be transferred and altered as desired.

Another method used is an edge detection algorithm followed by a least squares to average the black region into a single averaged point line to follow [14]. Applied but similar systems have been used to detect more vague but definitive lines such as rows in fields for agriculture seen in [15], this system uses a Hough transform to detect the more difficult lines found in fields.

Older methods that require more specified tracks use state machines to create a ground truth map by classifying edges as rising or falling based on the colour difference. Each possible state then has a defined response such as seeing only the left of the track (represented by +) the response is turn right [16]. Due to the fixed track segments as defined by NXP and the simple track setup, a similar approach may be taken to extract and respond with defined protocols.

More modern computer techniques can be applied for a novel small scale self driving car such as neural networks, the work in [17] uses these complex neural network techniques to train the control of a robotic hand to solve a Rubix cube. The motors in this to control the hand are synonymous with the motors to drive the car, the vision of the Rubix cube equal to the vision of the car and the touch feedback the same as reading real values back from the motors. Other solutions use the inbuilt computer vision technique cameras such as the NXP endorsed Pixy camera, this camera can be used to detect multiple line vectors and simply communicate these for steering and driving the car as seen in [10].

C. Steering/Path Tracking Controllers (David)

Path tracking controllers describe control systems which maintain a vehicle on a defined route. Current designs can be categorised under three broad types – geometric, kinematic and feedback based.

1) Geometric Controllers

Geometric controllers are based on the geometric relationships that can be found in generating arcs from a current point to a setpoint, and are one of the most popular methods for control due to their simplicity, fast cycle times and non-reliance on complex kinematic models [18]. The first geometric controller, pure pursuit, was developed in 1985 by Wallace et al. as the pathing controller for the Autonomous Land Vehicle project, and operates on the continuous process of pathing to a setpoint that is permanently a set distance ahead (known as look ahead distance L_{ad}), thus keeping the setpoint in perpetual 'pursuit' [19]. Pure pursuit is accurate, robust against a large error and fast to compute, but its accuracy and robustness is wholly reliant on L_{ad}, due to the variable dictating the distance at which set points are generated from. A look ahead distance too close causes oscillation about the path, due to arcs having to be 'drawn' at such a small length, while a longer look ahead distance results in smoother motion at the expense of losing accuracy and cutting corners of the route [18]. Variations have been made that aim to improve from a simple pure pursuit controller, such as using a variable look ahead distance based on speed or combining pure pursuit with a proportional-integral feedback. Yuanpeng Chen et al found that the implementation of proportional-integral feedback improved the steering responsiveness, further iterating upon this with a low pass filter to smooth the response by eliminating large changes in steering angle [20].

An alternative to pure pursuit was proposed by Stanford University, developing the Stanley method for their autonomous vehicle entry in the 2006 DARPA Challenge. The Stanley method implements non-linear feedback of the cross track error, defined as the distance from the front axle to the nearest setpoint on the route – this method, by using the cross-track error to determine setpoints as opposed to a look ahead distance, improves upon pure pursuit by avoiding corner cutting or oscilation from an incorrectly tuned look ahead

value, but can suffer from overshoot and is easily affected by noise due to direct error feedback [20] [21].

Vector pursuit is a complex path tracking technique that was developed by Jeffery Wit as a method to accurately control non-holonomic autonomous vehicles based on the theory of screws, which state that any motion of a rigid body can be modelled as a simultaneous lateral and rotational motion in and around the direction of movement respectively [22]. The method, tested on the Navigation Test Vehicle at CIMAR, was found to be less reliant on the chosen look ahead distance and extremely robust against large positional and orientational errors, such as when skirting an unexpected obstacle on the route [23].

2) Model Based Controllers

Model based controllers are designed around kinematic or dynamic models of the vehicle and give potentially more accurate control when modelled well, due to the controller having greater detail about the vehicle's behaviour than geometric or feedback controllers. Kinematic models often use smaller models such as such as a bicycle, and isolate the longitudinal and lateral dynamics so that the controller can determine a response for each term. Benoit Thuliot's controller and similar designs are effective and accurate at low speeds but struggle at higher speeds and more uneven conditions as the complexity of the vehicle behaviour increases, resulting in the kinematic model becoming less accurate – this trait however makes kinematic based controllers potent in applications such as parking and urban driving, where speeds are kept low and models remain accurate [24]. Attempts to eliminate the tracking inaccuracy at higher speeds have been made by developing models based on the dynamics of the vehicle, but are difficult to implement due to the complexity and high computational cost, displaying non-linear and discontinuous behaviour. Cheung Jun Ma et al. developed a model predictive control algorithm which used both kinematic and dynamic models, showing high robustness and small tracking error, but suffered from a slow response that limited its application [25].

3) Feedback Controllers

A type as common as geometric controllers, feedback controllers are easy to implement and have relatively low computational costs due to their non-reliance on complex models [20]. Controllers made by Pan Zhao [26] and Kapina [27] were robust and effective in minimising deviation from the path even at high speed and at vehicle handling limits. Therefore, due to their versatility and performance at high speeds, feedback controllers when turned correctly are effective solutions on their own, but are extremely successful when combined with other types of controller – the controller developed by previously mentioned Yuanpeng Chen demonstrated responsive, smooth and accurate control using PI feedback in combination with pure pursuit with variable look ahead distance, using the feedback controller to 'cover' the limitations of the pure pursuit when an incorrect look ahead distance was set [20].

V. Theory

A. Chassis and vehicle dynamics (Andrew)

The frictional force, F, available at the tyres is given by the coefficient of friction, μ, mass of the car, m, and acceleration due to gravity, g, in (1). It is worth noting that the proportion of total force at each tyre depends on the proportion of the total car weight on that tyre. Since a 2WD car only applies torque to two of the four tyres, the car only has half of the total force of the available to it to generate traction (assuming even weight distribution). A 4WD car has all the frictional force available to generate traction.

$$F = \mu m g \tag{1}$$

The tractive force needed to accelerate the car is a product of the car's mass and acceleration (Newton II). Equating this with the frictional force available at the tyres (1), gives the formula for the maximum acceleration, a, as limited by traction (assuming 4WD) forming (2). Note that the force in (1) would have been halved if applied to a 2WD chassis, resulting in a factor of 0.5 in (2).

$$a = \mu g \tag{2}$$

Cornering speed limited by grip at the tyres, v_c, for a corner with radius, r, can be determined by equating the equation for centripetal force of and object in a circular path with the frictional force available at the tyres (1), forming (3).

$$v_c = \sqrt{\mu g r} \tag{3}$$

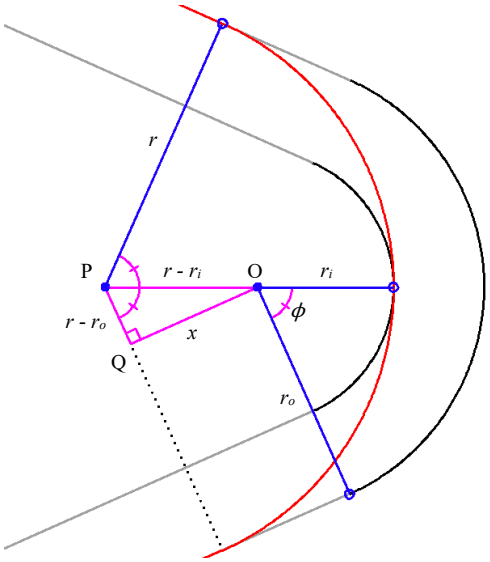

Fig. 1. Example curved (black) and straight (grey) track pieces with racing line (red). Key radii are shown in blue. The angle, ϕ, is half of the angle of the curved track piece. Points P and O, are the centres of the racing line and curved track piece respectively.

The radius of a racing line, r, through a curved track section (Fig. 1) can be found trigonometrically by observing the triangle $\triangle$OPQ and knowing the inner track radius, r_i, the outer track radius, r_o, and the half-angle of the track, ϕ (4).

$$\cos\phi = \frac{r - r_o}{r - r_i}$$

$$r = \frac{r_o - r_i \cos\phi}{1 - \cos\phi} \quad (4)$$

The racing line projects into the straight sections of track at the entrance and exit of the curve. This distance, x, can be calculated using (5) which can be simplified for the case of 90˚ bends (ϕ is 45˚) to (6), and for 180˚ bends (ϕ is 90˚) to (7) – the two types found in the race.

$$x = (r - r_o)\tan\phi = (r - r_i)\sin\phi \quad (5)$$

$$x_{90^\circ} = r - r_o \quad (6)$$

$$x_{180^\circ} = r - r_i \quad (7)$$

Using the Kinematic Equation (8), the maximum car speed, v_{max}, can be determined. This is based on the assumptions that on a straight the car will be accelerating (2) at traction-limited rate, a, for a displacement, s, and that in the corners the speed, u, will be constant and limited by grip (3).

$$v_{max} = \sqrt{u^2 + 2as} \quad (8)$$

Assuming the magnitude of acceleration and braking is the same and that the straight of length, l, starts and ends with two identicle corners, then the displacement is given by (9). The length of the racing line which extends into the straight is x (5).

$$s = \frac{l}{2} - x \quad (9)$$

The target speed and (force required for) acceleration can be translated in terms of angular velocity and torque at the motor shaft. These values are dependent on the adjustable gear ratio chosen for the connection of the motor to the drivetrain, as well as the one inherent in the drivetrain. These can be represented by a single gear ratio known as the final drive ratio (FDR). The FDR, R, depends on the tooth count of the motor shaft gear (pinion), T_p, the accompanying drivetrain gear (spur), T_s, and the drivetrain ratio, R_d.

$$R = R_d \frac{T_s}{T_p} \quad (10)$$

The motor shaft torque requirement, τ, can be calculated from the force, F, required for maximum acceleration (1), the wheel diameter, D, and the FDR, R in (11).

$$\tau = \frac{DF}{2}\frac{1}{R} \quad (11)$$

The motor angular velocity requirement, ω, can be calculated from the maximum car velocity, v_{max}, seen in (8), the wheel diameter, D, and the FDR, R, in (12).

$$\omega = 2\pi \frac{v_{max}}{\pi D} R = \frac{2v_{max}}{D} R \quad (12)$$

B. Motor and control (Andrew)

The car must be battery powered which limited the motor choice to DC types. Brushed DC (BDC) and brushless DC (BLDC) motors were considered.

1) Brushed and brushless DC motor comparison

BDC motors contain magnets in the stator which create a field over the rotor. The rotor contains coils through which the motor current is passed which make physical contact with the external supply through brushes and slip rings. The current in each coil is reversed every half-cycle by a mechanical commutator, meaning a unipolar current can be applied externally. This brings the benefit of simplified control, typically achieved using a PWM chopper circuit as part of an H-bridge. BDC motors also have a lower purchase price. However, the disadvantages are increased wear and frictional losses due to the brushes, as well as increased electrical noise from switching. The brushes also limit the maximum speed of the motor while adding a significant voltage drop, reducing efficiency. The inertia of the coils within the rotor is typically high as well [28].

BLDC motors remove the need for friction contacts and a mechanical commutator by placing the magnets in the rotor. The coils are found in the stator and each phase is available through an external terminal. Commutation is offloaded to a speed controller achieved using solid state switches, typically MOSFETs at the powers seen in this project. The lack of sacrificial brush contacts reduces wear and improves efficiency, while solid state switches prevent the electromagnetic noise emitted by sparking contacts. The lower inertia of the rotor improves dynamic response meaning increased acceleration. Although controller complexity is increased, commutation has the potential to be varied in response to the motor speed and torque. This leads to potential increases in control precision and efficiency [28]. The simplest switching type for BLDC motors is trapezoidal and is typical at the model car level.

2) BLDC motor characterisation

Typical hobby BLDC motors are specified by their rated current, maximum voltage and k_v constants. k_v takes the units RPM/V and is inversely proportional to the back-EMF constant, k_e, measured in V/rad. Both constants are measured phase to phase and once in SI units, k_e is equal to the torque constant, k_τ, as seen in (13) [29] [30].

$$k_e = k_\tau = \frac{60}{2\pi}\frac{1}{k_v} = \frac{9.5}{k_v} \quad (13)$$

A BLDC motor with ideal trapezoidal commutation is equivalent to a BDC motor and so the same equations apply [30]. The angular velocity, ω, and electromagnetic (no loss) torque, τ, are therefore given by (14) and (15) respectively,

where V is the peak phase to phase voltage and I is the DC supply current [29].

$$\omega = \frac{V}{k_e} \tag{14}$$

$$\tau = k_\tau I \tag{15}$$

The no load speed can be calculated from (14) using the maximum motor voltage specified. The rated torque can be calculated using (15) and the rated current specified. The maximum current is generally not stated by hobby motor manufacturers which complicates finding the stall torque and plotting the speed-torque curve. From observation of the few hobby motors where the winding resistance was specified, the maximum current calculated at maximum voltage was seen to be typically 10 to 20 times that of the rated current. This estimate was used in the project to calculate stall torque.

3) Sensored and sensorless commutation

In order to commutate the motor at the correct time, the controller requires feedback to determine the angle of the rotor. This can be done either using sensors within the motor or sensorless by measuring back-EMF.

Sensors are typically either optical encoders or hall effect sensors. Optical encoders give a high precision reading of the exact rotor position, allowing for more advanced commutation strategies such as sinusoidal, but are typically more expensive. Hall effect sensors give a binary output in response to the presence of the magnetic field generated by the rotor. Typically positioned at 120˚ intervals, the sensors give a lower resolution, and are best suited to simpler trapezoidal commutation. Hall effect sensors are seen on less expensive motors. Sensored control is typically simpler from a controller perspective and ensures smooth torque at low speeds. However, the sensors increase motor cost and add additional failure modes [28].

Sensorless control is most commonly implemented by measuring the zero-crossing point (ZCP) of the back-EMF generated by the rotor. For trapezoidal commutation, phase current is started after a 30˚ delay from the phase ZCP and stopped 120˚ later. Sensorless control increases controller complexity but removes the additional cost and failure modes associated with motor-based sensors. A significant disadvantage of sensorless control is that induced back-EMF is impossible to detect at low speeds, resulting in open loop control. This results in torque fluctuations until the motor is at a sufficient speed to induce measurable back-EMF [28] [31].

4) Braking

Just like BDC motors, BLDC motors can operate in any of the four quadrants and can be made to impart a braking torque by the controller. The main braking modes are dynamic, plugging and regenerative. All attempt to manage the reverse current generated when the motor voltage source is less than the back-EMF of the motor [32].

Dynamic braking disconnects the source, essentially setting it to zero, and recirculates the reverse current through the windings by shorting the motor phases either directly or through a dump resistor. This form of braking is poorly controlled as the braking torque is dependent on the current flowing, decaying to zero at low speeds. The excess energy is wasted, generating significant heat [32].

Plugging involves reversing the polarity of the voltage source supplying the motor during braking. This increases the braking torque by essentially attempting to reverse the motor and results in controllable deceleration and torque down to zero speed. However, the reverse current is increased massively which increases the power dissipation and heat generated. This also wastes additional energy from the source imparting a braking torque [32].

Regenerative braking keeps the voltage source connected, causing the reverse current to flow back towards it. Although under normal conditions the back-EMF will not exceed a fixed voltage source, the PWM chopper circuits used to drive the phases and the inductance of the stator coils act as a boost circuit to ‘force’ the flyback current into the supply during the PWM on-phase. For unidirectional supplies, the regenerative current can be a hazard and external bypass circuitry is required to safely dissipate the energy. However, for battery powered motors, the regenerative current can be harnessed to re-charge the battery. As well as being the most efficient braking method, regenerative braking can be controlled by varying either the supply voltage or the switching of the phase chopper circuit [32].

Alternatively, the MOSFET body diodes in the phase chopper circuits can be used to drive the motor in discontinuous current mode. The diodes prevent reverse current from building. Instead, during the PWM off-period the current falls to, and stays at, zero. During the on-phase, the supply voltage is larger than the back-EMF so a small current flows in the forwards direction. This results in no braking torque being applied when the supply voltage is less than the back-EMF, allowing the motor to ‘freewheel’ (not to be confused with freewheel current caused by motor inductance).

C. Steering Servos (David)

1) Overview of mechanics

Servomechanisms, or servo-motors, define devices which control and regulate the speed and position of a load. While the true origins of the term are unknown, the concept of a servomechanisms has been prevalent far before the first coinage of the term in 1868 by Farcot, to describe hydraulic and steam engines designed for ship steering [33], with mechanisms to position windmill sails so that they always faced the wind [34]. Mechanically, modern servos consist of a DC motor, typically brushless due to their higher speeds, greater reliability and longer operating life compared to brushed [35], with gearing to translate its high RPM to torque. Control of the servo motor is done by a positional controller, amplifier, and feedback sensors such as a potentiometers, tachometers, resolvers or rotary encoders. Servos therefore are closed-loop systems that receive a set point via a command signal, coordinate the required response from the positioning controller, and amplify the response from the controller to power the motor and move the load, with the result being monitored using the feedback sensor to correct errors (see Fig. 2) [36].

Servomechanisms, or servo-motors, define devices which control and regulate the speed and position of a load. While the true origins of the term are unknown, the concept of a servomechanisms has been prevalent far before the first coinage of the term in 1868 by Farcot, to describe hydraulic and steam engines designed for ship steering [33], with mechanisms to position windmill sails so that they always faced the wind [34]. Mechanically, modern servos consist of a DC motor, typically brushless due to their higher speeds, greater reliability and longer operating life compared to brushed [35], with gearing to translate its high RPM to torque. Control of the servo motor is done by a positional controller, amplifier, and feedback sensors such as a potentiometers, tachometers, resolvers or rotary encoders. Servos therefore are closed-loop systems that receive a set point via a command signal, coordinate the required response from the positioning controller, and amplify the response from the controller to power the motor and move the load, with the result being monitored using the feedback sensor to correct errors (Fig. 2) [36].

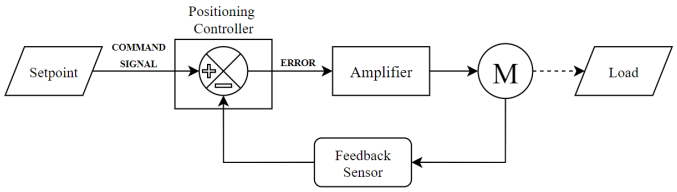

Fig. 2: Typical servo system diagram

Servos can be classified under three overarching types: positional rotation, continuous rotation and linear. Positional rotation servos essentially act as levers, with a single output shaft with a rotational range of 180 degrees, and its PWM control signal interpreted as an angle setpoint – this type of servo is common in remote control hobbyist applications such as rudder control for boats or wheel steering for cars where the range of motion required is finite. Continuous rotational servos function more similarly to standard motors, allowing rotation in either direction indefinitely by taking its PWM control signal and interpreting its pulse length as speed and direction of motion instead of a position – these servos are seen in applications that can utilise the full range of motion this type grants such as control of a radar dish or as a cheaper drive motor than stepper or standard DC motors. Linear Servos are a variation of positional rotational servos, converting its rotation to a forwards and backwards movement using a rack and pinion gear, and are commonly utilised in systems which require controlled lifting power greater than what a positional rotation servo could provide, such as robotic arms.

Servo complexity can significantly increase based on the level of precision required such as backstepping and fuzzy logic in robotics and industrial systems [37] [38] [39]. RC Servos used in this paper however use simpler variations of PID, commonly proportional-integral plus bang-bang control [40], due to the relatively lower precision demands for RC hobbyist applications.

2) *Servo Characterisation*

Due to the system being battery powered, the choice of servo was restricted to DC powered servos. Selection was chosen on categorisation based on its signal processing, speed, torque, and size characteristics.

a) Analogue vs Digital Servos

In both analogue and digital systems, the input voltage is received as a PWM signal with a frequency of 50Hz and passed through a pulse width to voltage converter. This conversion enables comparison between the current and reference voltage for feedback. From here, analogue and digital servos differ by the way they process their input signal and feedback.

In analogue systems, feedback is commonly determined using a potentiometer in which its contact is directly integrated into the output shaft, resulting in a variable voltage as the shaft turns. This variable voltage can then be fed into an error amplifier, typically an op-amp with negative feedback [41], which compares between the input and potentiometer voltage to output an error voltage that is passed to an amplifier and then to the motor, continuing this cycle until the error voltage is zero and the servo is at the correct position. Analogue servos are inexpensive, consume a lower amount of power than digital and are more scalable, with larger industrial servos commonly being analogue. However, analogue servos suffer from positional tracking errors, an imprecise zero point and shorter lifespan due to the use of a potentiometer as a feedback sensor being affected by temperature and power supply variation as well as physical contact wear [42]. Additionally, as the response powering the motor is proportional to the error, analogue servos struggle with generating sufficient torque to move the motor on small movements. This problem is remedied by assigning a minimum pulse length, known as to hobbyists as deadband, but this reduces the precision, or resolution, of the analogue servo due to this minimum pulse length equating to the smallest movement possible. This combined with output pulses only occurring at 50Hz means its response is more uneven and outputs a lower torque-weight ratio compared to digital servos.

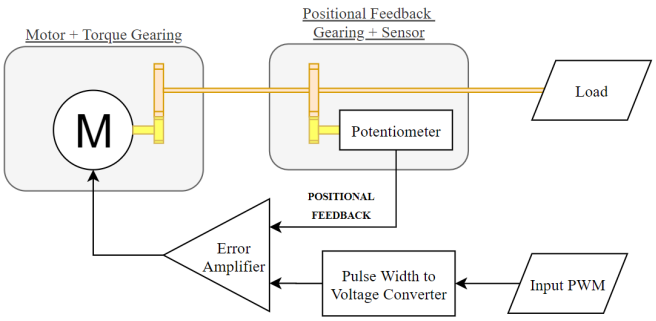

Fig. 3: Circuit diagram of a typical analogue servo, using a potentiometer directly connected to the output shaft as its feedback sensor

Digital servos differ from analogue as the implementation of a microcontroller changes the processing of the input signal - by taking the pulse length and determining the required response, the microcontroller is then able to continually output to the motor in smaller, much more frequent pulses. As response time and torque are proportional to pulse frequency, the significantly faster pulses provided by the microcontroller result in a much faster response, greater torque-weight ratio versus analogue servos, as well as a higher degree of precision

due to the smaller pulses contributing to a much smaller deadband. Due to its microcontroller, digital servos are also reprogrammable through systems such as Arduinos, allowing configuration of parameters such as its pulse frequency and deadband to optimise its performance. Overall, digital servos are usually superior to a similarly rated analogue servo, at the expense of greater power consumption and a higher price.

b) *Speed, Torque and Size Characteristics*

RC servos are produced with defined sizes, ranging from nano, weighing less than 7g for usage in micro-planes, to 80g and greater large servos for giant scale planes and jet flaps, with the competition car using standard size servos.

Servos can also be categorised via specialised models that maximise the performance of the servo in a certain parameter. High voltage servos are powered by voltages of 7.4V and above (as opposed to standard servos which operate from 4.8-6V) and are used in larger scale RC vehicles, as well in systems with higher voltage power supplies that are unable to use a battery eliminator circuit. The higher voltage input allows for greater torque and speed than standard servos, allowing a trade-off between performance versus higher cost and greater power consumption.

High torque servos provide greater torque per volt, outputting a torque of at least 10kg-cm (0.98N-m) at their minimum input voltage. These are essential in applications such as heavy robotics or off-road RC cars where the additional power is needed to lift loads or maintain steering control in uneven conditions.

D. Computer vision (Hamish)

In this project most of the control problems faced are around detecting the racing cars environment, such as detecting the track lines, intersections and speed limit signs. Many of these tasks are completed daily almost sub-consciously at a much higher complexity by humans whilst driving, purely by sight. As such a system that could partially mimic the human visual system (HVS) and output control signals would drive the car in a similar fashion to that of a human. This definition of mimicking the HVS is generally called computer vision and isn't a purely engineering field given some of the most influential papers were through biological studies [43]. Computer vision, in engineering terms, is usually seen as a subsection of artificial neural networks, designed to mimic the human brain [44]. As such it's important to understand how such systems function.

1) Feed Forward Neural Networks

The most basic and common neural network is called a feed forward network [45], seen in Fig. 4. In this many variables (Neurons), represented by circles are connected by weighted functions (Synapses). The column of neurons on the left of the diagram is known as the input layer as this is where the input data enters the neural network, in the case of a video this would be pixel colour values. The next two layers are known as hidden layers, each Neuron in the network is connected to every neuron in the previous layer with a trained function (Typically a single multiplicative value assigned to the connection). This is commonly referred to as a dense neural network given the dense connection jungle. These layers are known as hidden layers as during typical operation, these values are never used and mean very little in the whole scheme and thus are never used externally, creating a black box system. Finally, the right-hand column is known as the output layer where each neuron has a specific meaning, sometimes analogue and sometimes binary such as is this row of pixels a part of a track, or what are the chances that this row of pixels is part of a track?

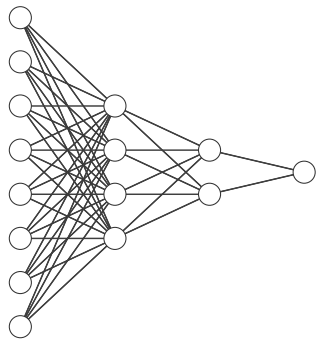

Fig. 4 Basic feed forward network with eight inputs, two hidden layers of four and two neurons respectively and a single output. Note this diagram is purely an example and the connections (Synapses) are unweighted.

These synapse connections can be trained in many ways but all teaching systems belong to one of three groups, supervised [46], unsupervised [47] and reinforcement learning [48]. The main difference between these methods are the input data required, supervised learning requires both the input and output data such as for a classifier an input photo of a shoe and the label output "Shoe". For an unsupervised system however, the network relies only upon input data and clusters data based on similarities meaning data doesn't need to be classified to be processed which is very useful for large sets of data. Finally reinforcement learning which requires the same input data as well as some rating of completion (Reward). Supervised learning for a self driving car would require input visual data and the corresponding movement such as the steering angle and speed to then train to the same proficiency as the human driver. Unsupervised training would need to take a different approach as the output isn't as simple as a direction and speed, instead the network would be trained to classify and group the track it sees these clusters could then be assigned labels such as corner and straights and some defined action taken. On the other hand reinforcement learning has the ability to out-perform a human driver as it is not limited by the test data labels and can still output simple control data, unfortunately an unsupervised would likely need to be simulated given as soon as the car veers off the track the car would need to be placed back on the track and given a low rating. A semantically similar issue is solved with reinforcement learning through simulation to solve a Rubix cube with one robotic hand to then be used tangibly [17].

Given the huge amount of data the HVS processes every second, around $10^8 – 10^9$ bits [49], these dense networks, even with modern computers, cannot train or even run these huge networks. Even a RGB Full HD camera (1920x1080px) would require at least 6220800 input neurons each needing to be connected to the following hidden network neurons, you can

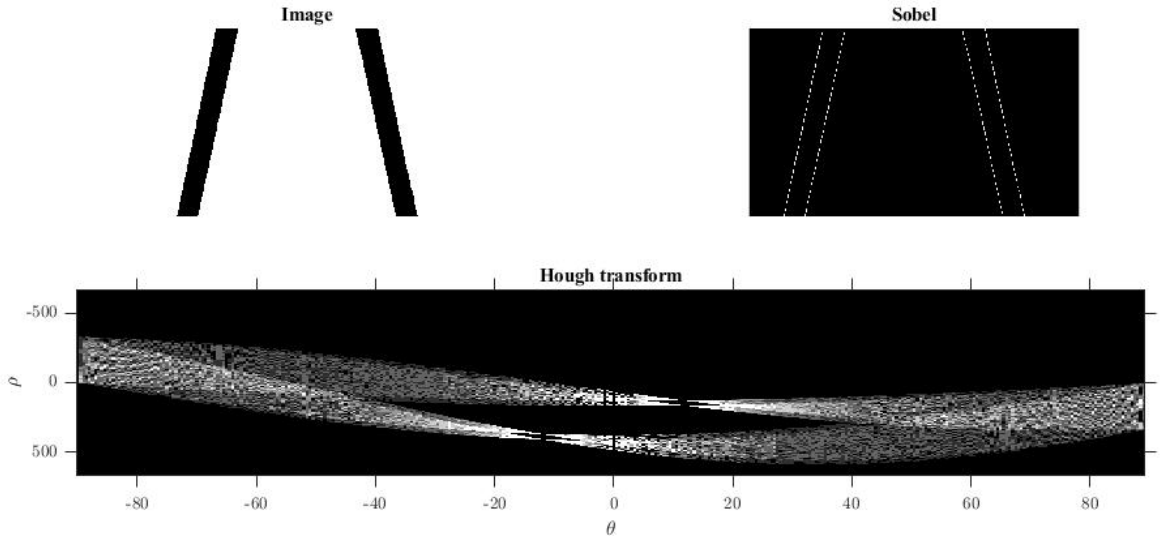

Fig. 7. A likely racing track image Sobel filtered to retrieve edge information passed into the θ- ρ domain. The Hough transform shows two groups of similar intersections with one group positively angled and another negatively angled both with close proximity due to close ρ values. This describes both the left and right edge of the left and right track showing the ability to classify the edges by Hough transform.

start to imagine the combinational explosion in dealing with these dense networks for images. Due to this data explosion when dealing with visual data, techniques such as convolution and max pooling need to be deployed to reduce such data down to reasonable levels.

2) Convolutional Neural Networks

To reduce the amount of data processed, CNNs (Convolutional Neural Networks) are used [50]. CNNs also have input, hidden and output layers like a feed forward network but make use of three key image processing processes, convolution, max-pooling and dense neural networks as seen in Fig. 6.

a) Convolution

Convolution is the process of convolving (dot product) an image with a kernel matrix, the output of this convolution is a feature extraction map where the features depend on the kernel used. These kernels are usually randomly selected and trained along with the neural network to extract the most necessary feature maps. The feature maps generated are an abstraction from the real image, such as a Sobel [51] style edge detection, many more feature maps are generated to extract as many features as possible as seen in Fig. 6. Convolution by nature, reduces the amount of data stored but creates many more feature maps meaning data isn't always reduced, instead pooling (almost always max) is used to extract the most important convolutional data.

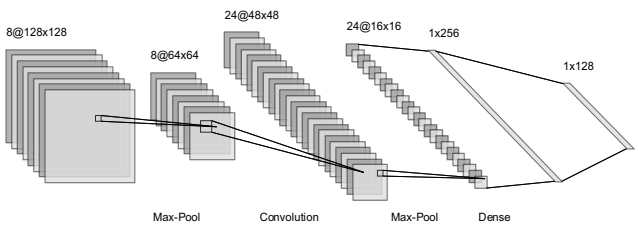

Fig. 6. *LeNet* [66] *style convolutional neural network with alternating max-pooling and convolution. The last two stages are dense neural networks formed from the flattening of the convolved images..*

b) Max pooling

Max pooling, unlike convolution, does not extract feature maps and as such does not change the number of images, instead pooling is used to compress image data whilst keeping the most important parts. Max-pooling works by moving a kernel over an image and selecting the maximum value in the kernel to carry over into the next image, this means for a 2x2 kernel the image data is reduced by 4 times. Using this max-pooling technique versus average pooling leads to no lack of sharpness in the image which is very important for edge detection and feature maps natural to the HVS [43].

c) Dense neural network

Dense neural networks, as described above, are all connected nets and the main stage of any neural network. To be able to apply a dense neural network to an image requires the data extraction and compression from convolution and pooling techniques to be one dimensional (a list) this is achieved by flattening the feature maps similar to how an image is stored in memory. From that point, image data can be used identically to that of any feed forward net to be trained as previously described.

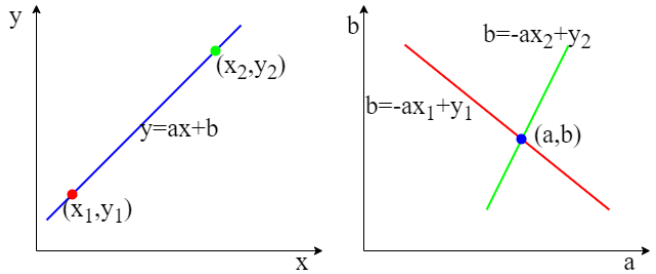

Fig. 5. A representation of two-point conversion between the x-y co-ordinate point space into a-b feature/line space

3) Hough Transform

The Hough transform is an image processing algorithm used to detect and quantify straight lines [52]. The Hough transform takes an image already processed by an edge detection algorithm such as the Sobel kernel [51] transforming the x-y plane edge co-ordinates into a-b feature space possible gradient and position lines as displayed in Fig. 5. The generated lines in a-b space are then translated into θ- ρ defined by the line angle and radius from the origin. For an image with many edges a map of all points θ- ρ can be made as seen in Fig. 7. This figure displays high likelihood lines in white and low in black with the intersections being most likely lines.

4) Optical Flow

Optical flow is a method of mapping the apparent motion of objects between video frames [53]. Optical flow is used in many modern applications such as drones [54], self-driving cars [55] and more. Optical flow works by taking assumptions between frames such as objects keeping their color and intensity leading to the following equation for a pixel at (x,y) at time t.

$$I(x,y,t) = I(x+\Delta x, y+\Delta y, t+\Delta t) \tag{16}$$

Using Taylors theorem [56]

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t) + \frac{\delta I}{\delta x}\Delta x + \frac{\delta I}{\delta y}\Delta y + \frac{\delta I}{\delta t}\Delta t \tag{17}$$

Cancelling to

$$\frac{\delta I}{\delta x}\mathrm{V_x} + \frac{\delta I}{\delta y}\mathrm{V_y} + \frac{\delta I}{\delta t} = 0 \tag{18}$$

And therefore

$$I_x \mathrm{V_x} + I_y \mathrm{V_y} = -I_t \tag{19}$$

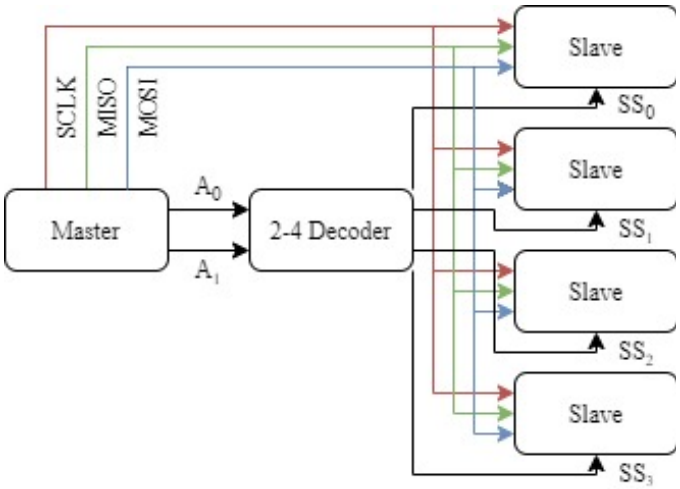

Fig. 8. A four un-cooperative slave SPI communication example using a 2-4 decoder for five wire control.

Giving the x and y velocity of the pixel (x,y) given the image derivatives (I_x, I_y and I_t). This is still unsolvable given two unknowns and requires more assumptions, the most common solution is the Lucas-Kanade method [57].

a) Lucas-Kanade Method

The Lucas-Kanade method assumes that neighbouring pixels have a similar motion, grouping pixels into average moving blocks. Doing so allows for the eight neighbouring pixels and the centre pixel to calculate their (I_x, I_y and I_t) values. As only ($\mathrm{V_x}$, $\mathrm{V_y}$) is required, Lukas-Kande method uses least squares to calculate a more accurate average to compute:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x q_i^2 & \sum_i I_x I_y q_i^2 \\ \sum_i I_y I_x q_i^2 & \sum_i I_y q_i^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x I_t q_i^2 \\ -\sum_i I_y I_x q_i^2 \end{bmatrix} \tag{20}$$

Once knowing the velocity of pixels in an image, it's possible to use this information to calculate distances, object and average camera velocity and acceleration.

E. Inter-board communication (Hamish)

An important factor in data processing and peripheral control is inter-board communication. The limitations of communication methodology can lead to many issues and therefore need to be compared before implementation.

1) Inter-Integrated Circuit (I^2C)

I^2C, now owned by NXP, is a communication protocol that uses two wires, Serial Data (SDA) and Serial Clock (SCL) driven in open collector mode thus requiring pull up resistors. Given the single data line I^2C is intrinsically half-duplex meaning one directional communication at a time. As standard I^2C is 100kHz making it one of the slower protocols yet less noise susceptible. I^2C slaves are enabled through the standard strict communication protocol meaning chip select lines aren't needed. The number of slaves available on a I^2C bus is defined by the address size, usually 7 bit and therefore $2^7 = 128$ devices excluding a small set of reserved addresses.

2) Serial Peripheral interface (SPI)

SPI , now owned by NXP, is a communication protocol that uses three wires, Serial Clock (SCLK), Master Out Slave In (MOSI) and Master In Slave Out (MISO) allowing for full-duplex communication. For communication with a SPI device a Slave Select (SS) pin is required for each device connected to be told when it is being communicated with. This means for eight independent devices each device requires connection to the three data lines and a separate SS pin meaning 11 connections are required. This can be lowered by using a three to eight decoder thus requiring six connections similar to that seen in Fig. 8 or by having co-operative slaves using the same SS line. SPI is loosely defined in terms of standards, but the most common communication speed is 2MHz usually definable up-to 10MHz but sometimes up to 100MHz.

3) Universal asynchronous receiver-transmitter (UART)

UART, unlike the previous, isn't technically a protocol, instead a communication device for interoperating serial or

parallel data. UART systems require one wire for serial communication, Transmit (TX) and Receive (RX) depending on if sending or receiving data and thus two wire for full-duplex data. For parallel communication the same TX and RX labels are used for the devices but as many wires required as parallel bits on the bus. UART requires both devices to have defined communication protocols defining the bit rate and length. UART is therefore used for one to one communication and cannot address multiple devices using the same connections without external hardware and design.

F. Steering Control (David)

While timing constraints meant that a PID feedback controller was used, the intended method was pure pursuit due to the computational cost reductions due to the avoidance of derivative terms which can cause a slower system [58].

Pure Pursuit is based around the geometric relationships depicted in Fig. 9. Here, the goal point (g_x, g_y) is set at the look ahead distance L_{ad} from the rear axle, and from these two points the steering angle δ can be calculated by mapping an arc is that passes through the rear axle and goal point:

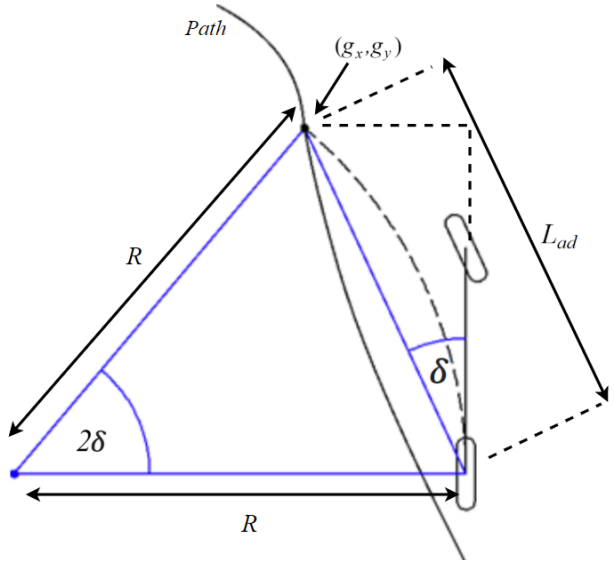

Fig. 9: Pure pursuit geometry

Applying the law of sines to wheel angle δ and arc angle 2δ (21), substituting $\sin(2\delta)$ and $\sin(90-\delta)$ with trigonometric identities (22), and simplifying and rearranging for R gives (23).

$$\frac{L_{ad}}{\sin(2\delta)} = \frac{R}{\sin(90-\delta)} \tag{21}$$

$$\frac{L_{ad}}{2\sin(\delta)\cos(\delta)} = \frac{R}{\cos(\delta)} \tag{22}$$

$$R = \frac{L_{ad}}{2\sin(\delta)} \tag{23}$$

As the curvature of an arc, κ, is defined as $1/r$, substituting this into the radius equation gives (24). Additionally, the steering angle δ can be defined as (25) by referring to Fig. 9, where L is the adjacent length of the triangle set by the steering angle δ (the edge including the vehicle length).

Fig. 9: Pure pursuit geometry

$$\kappa = \frac{2\sin(\delta)}{L_{ad}} \tag{24}$$

$$\delta = tan^{-1}(\kappa L) \tag{25}$$

Finally, by substituting the curvature (24) into the steering angle equation (25), the steering angle of the front wheel and thus the control law is derived [59]:

$$\delta(t) = tan^{-1}\left(\frac{2L \cdot \sin(\delta)}{l_d}\right) \tag{26}$$

G. Sonar distance measurement (David)

Ultrasonic sensors calculate distance by using the reflective properties and relative consistency in the speed of sonic waves, measuring the elapsed time between the emittance of a 'trigger' ultrasonic wave and its reflection returning.

Ultrasonic waves are generated via an ultrasonic transducer. By applying an alternating electric field to the piezoelectric material, sonic waves are forced out from the transducer via the pressure created by the rapid expansion and contraction of the material. This effect can be reversed, with ultrasonic waves absorbed by a transducer inducing an electric field, and this reversable trait is used to create a sensor by mounting two transducers together as a transmitter and receiver.

To take a measurement, an ultrasonic burst of defined length is sent by the transmitter to generate an 'ultrasonic signature' that can be used to identify the pulse from ultrasonic background noise. The receiver (echo) pin is then set high to mark the start of the return echo and is held high until the return signal is received, or the defined timeout period has elapsed. On successfully receiving a return signal, the period between the transmitter pulse and the echo pin going low dictates the time taken for the signal to travel to the object and back.

The distance from the sensor to a body can be defined as (27), where T_f is the time taken for the ultrasonic wave to travel distance D, k is a constant based on the sensor geometry (~0.5), and V_s is the velocity of ultrasound waves. V_s is predominantly affected by temperature and can be approximated by (28), where T is the absolute temperature in Kelvin.

$$D = kT_fV_s \tag{27}$$

$$V_s = 20.055\sqrt{T} \tag{28}$$

The choice of frequency can affect the performance of the sensor. Higher frequencies provide a greater resolution for the greater cost in using faster transducers and reduced range due to greater attenuation in the air. Lower frequencies benefit from lower scattering of the wave on reflection and a lower price to transmit due to needing less expensive transducers, however, longer wavelengths in low frequency sonic waves make distinguishing the echo pulse and ultrasonic background noise more difficult, risking unreliability – approaches have been

made however to circumvent this by implementing methods of digital signal processing to help differentiate the echo pulse [60] [61].

VI. Implementation

This section provides an overview of design and implementation of key project sections. An overall block diagram of the design is shown in Fig. 10. The final car specifications can be found in Appendix TABLE III.

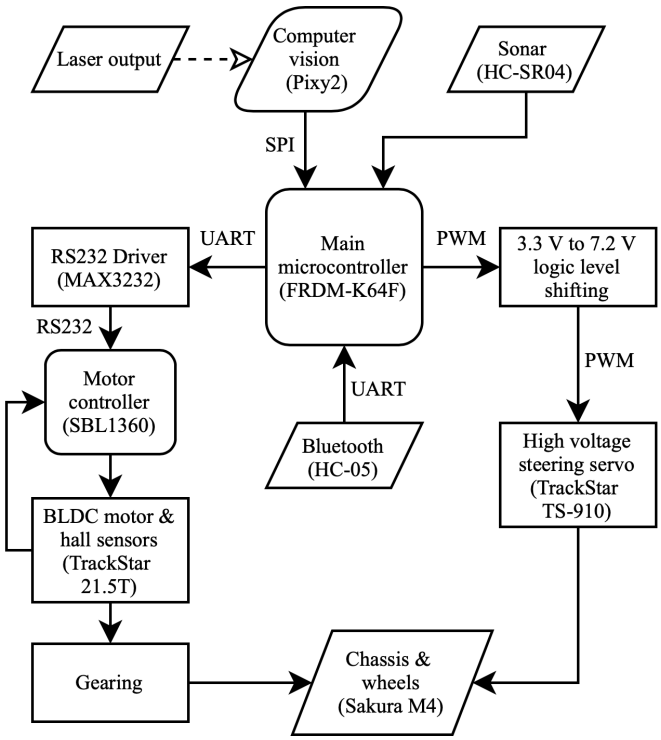

Fig. 10. (Andrew) Block diagram of overall system design. System input/output blocks in parallelograms, blocks containing adjustable microcontrollers are rounded. Laser has been purchased but not assembled, sonar has been interfaced with microcontroller but not tested.

A. Chassis, Drivetrain and Fixings (Andrew)

Designing a functional car which would not limit the performance possible with the control systems on board was critical to achieving the objective to design a car and the aim of winning races. Therefore, significant time was spent estimating and choosing components.

1) Chassis

Although a chassis recommended by NXP (including motor, speed controller, steering servo) had been back-ordered by the University, it was decided that a superior one should be chosen as part of the project objectives. There was no indication of when the back-ordered parts would arrive. The kit had a two wheel drive (2WD) setup, with two motors directly driving each rear wheel. As already explained, 2WD would limit the acceleration and braking and potentially interfere with steering due to discrepancies between torque produced by each motor. Two motors would also require two controllers which would increase cost and weight, especially if more advanced controllers were used.

The main limitation when choosing the chassis was that the wheelbase should be short enough to navigate the track specified by NXP. Although no specific limits were set, the wheelbase of the recommended kit-car was 180 mm. Most racing-grade model cars are in the $1/10^{th}$ scale category and have a 255 mm wheelbase. Reducing this as much as possible would reduce the car's tuning circle and allow more freedom in navigating the track following a 'racing-line'.

Four wheel drive (4WD) was another crucial chassis requirement to maximise acceleration and braking and was the main factor in limiting selection to the $1/10^{th}$ scale racing category. The category also offered damped, tuneable, independent suspension as well as wheel camber (vertical angle) and toe (horizontal angle) adjustment. As well as the increased stability given by the dampers, the wheel angle adjustments would allow for a higher degree of optimisation to match the track requirements if desired in future.

The final requirement was for differentials on the driving axles. These would reduce slip during turns and hence increase traction when cornering and so were reasonably important. Furthermore, despite not being an objective, they would allow for the introduction of a torque vectoring system if the car was used in future years. Torque vectoring would be simpler to implement with the kit car as the rear wheels were independently driven, however the benefits of 4WD would significantly outweigh those of torque vectoring. No such system was available in model racing which exclusively used single motor drive. All 4WD $1/10^{th}$ scale cars did however contain at least a rear differential, with many also featuring a front differential. This would allow for a torque vectoring through braking system so long as brakes could be sourced. Brakes were available but generally for larger chassis and would have to be modified to fit a $1/10^{th}$ scale chassis.

The chassis chosen was the 3Racing Sakura M4. This featured an adjustable wheelbase down to 215 mm setting it aside from the majority of $1/10^{th}$ chassis. It also had belt-driven 4WD, a rear differential and all the aforementioned ride adjustment options. While the lack of a front differential meant the front wheels would experience additional slip, the design meant a second rear differential could be purchased and fitted to the front axle if later required. As torque vectoring was a secondary objective this was a reasonable compromise for a shorter wheelbase. The chassis also featured standardised motor, servo and body mounts, simplifying assembly and design of an electronics mounting board.

2) Motor and gearing

Using (3) through to (9) the maximum car speed for three probable scenarios was calculated. These scenarios all contained a straight section 4.14 m in length, based on the longest straight in the example track. At the entrance and exit of the straight were either two 90˚ bends or two 180˚ bends. These would create two different racing lines. The third scenario was no racing line, i.e. the car followed the centre of the track. The difference in angle of the two corner types would have no effect in this case. The coefficient of static friction was assumed to be 0.9, a commonly used value of rubber on tarmac. Track dimensions were taken from [8]. The results are in TABLE I.

The highest speed came from following a centre line. This was expected as the idea of a racing line is to increase average speed rather than maximum speed. Looking at the distance travelled in a straight line, s, it is clear that the racing lines extend into the straight, reducing the distance over which the car can accelerate - limiting top speed. Though the corner the car must maintain a constant speed limited by the grip available. It can be seen that the racing lines increase this corner speed, v_c. The drivetrain was chosen to reach this worst case top speed while maintaining maximum acceleration up to it.

TABLE I
CAR SPEED AND ACCELERATION ESTIMATES

Scenario	a (ms^{-2})	r (m)	v_c (ms^{-1})	s (m)	v_{max} (ms^{-1})
Racing line 90˚	8.82	2.05	4.25	0.742	5.58
Racing line 180˚	8.82	0.720	2.52	1.52	5.76
Centre line	8.82	0.44	1.98	2.07	6.36

Calculation of maximum car speed, v_{max}, maximum acceleration, a, radius of corner as driven, r, speed through corner, v_c, and distance to middle of straight as driven following a corner, s.

Since the car chassis was already designed for standard hobby grade motors and due to their availability and price, hobby motors were the only ones considered for the project. The manufacturer recommended motor specifications were used as a starting point [62]. Similar motors were characterised and compared to the required load to finalise selection. As well as specifying k_v, hobby suppliers frequently use the stator coil turn count to categorise motors, which is inversely proportional to k_v. The recommended motor turn count was 13.5, so more readily available motors above and below this range were characterised using (13) through to (15), based on the assumption that stall current was 10 times rated current as previously explained (Fig. 11). A load line of required speed and torque was also calculated at various FDRs using (10) through to (12).

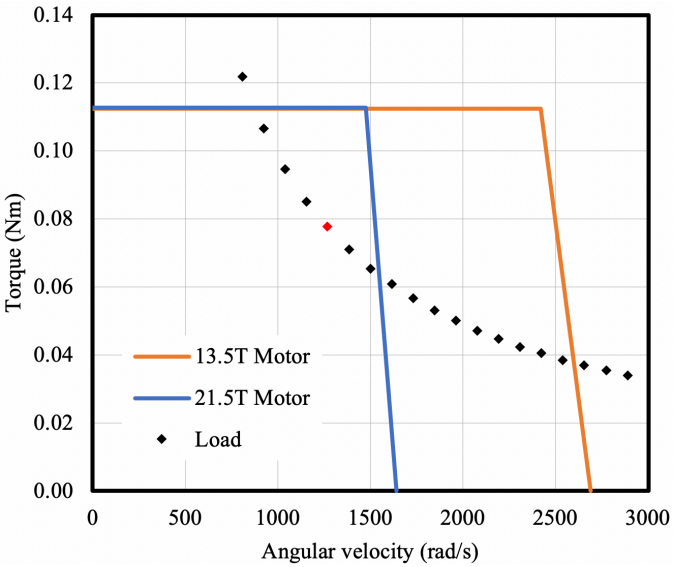

Fig. 11. Motor speed-torque curves for two most probable motors. Maximum set as rated torque. Load curve of car as seen at the motor shaft with varying final drive ratio. The red point is the one achieved with the initial gear choice.

From Fig. 11, the 21.5 turn motor was chosen. The red point was the load FDR closest to the pinion and spur gears supplied with the chassis. It was decided that the 13.5 turn motor would be too fast, essentially limiting the controller output to 50% and potentially reducing control precision.

The characterisation did not include any losses, nor was inertia of the transmission included. This was due to lack of data about the resistance of the motor, as well as lack of data about mass and dimensions of the rotor and of the transmission. However, the angular velocity potential of the motor was still 20% greater than required and the torque was 45% greater than required, leaving reasonable margin for error. Furthermore, the FDR could be inexpensively varied by purchasing a different pinion gear if only one of the requirements was lacking.

The motor was a sensored type, using three hall effect sensors designed for trapezoidal commutation. As previously explained (p6), sensored commutation offers smoother control with more torque at low speeds. Since the car has significant mass, this could result in jerking at low speeds, reducing initial acceleration and unnecessarily wearing the drivetrain. As a result, sensorless control is rarely used in hobby racing grade cars.

3) Controller and cell chemistry

The motor controller requirements were that it must be capable of sensored trapezoidal control, controlled braking, torque limiting and closed loop speed control. The type of commutation was fixed by the motor choice. Braking was essential to achieving quick lap times as otherwise the top speed would have essentially been limited by the speed carried through the corner with the smallest radius. Torque limiting was required to prevent wheel spin on heavy acceleration and braking since the motor was selected to have enough torque to break traction provide maximum acceleration through straight sections of track. Closed loop speed control was necessary ensure the car could reliably run at the upper limit of speed while maintaining traction through corners. As slip detection was not being implemented, speed was the most effective control variable to maintain grip.

The most readily available controllers were targeted at the hobby racing market. While these were inexpensive and rated for the currents and voltages likely to be seen with a battery-powered car, the documentation with them was lacking. Very few appeared to offer closed loop speed control, meaning an external speed sensing and control device would need to be built and tuned. Although linear controllable braking was commonly available, the type of braking was not specified, nor was the exact response. Most probably, the braking would have been plugging based since it should have been effective to zero speed and regenerative braking is rarely seen at these price points. Plugging may have resulted in excessive heat generation given the additional mass of the car from the additional electronics it was not designed to carry. The controllers also offered ‘freewheeling’ i.e. if the input ‘throttle’ was reduced, rather than generating negative torque, the motor would roll until frictional forces slowed it down, as explained previously (p6). Instead, braking had to be explicitly specified as a separate command. This non-continuous input would complicate the development of an external controller.

Furthermore, the controllers only offered rudimental torque limiting in terms of an arbitrary variable to limit the rate of change of 'throttle'. It was not clear how reliable this would be, nor whether it also applied to braking.

Instead of hobby controllers, commercial industrially targeted ones were investigated. A major obstacle was that these greater precision controllers all required higher operating voltages than what could be supplied by hobby Lithium Polymer (Li-Po) batteries. Li-Po was the preferred choice due to the compatibility of its standardised, pre-assembled hobby car form factor with the chassis chosen. The chemistry also offered excellent power to weight and capacity to weight ratios. However, the competition rules limited Li-Po batteries to two cells, producing 7.4 V nominal, less than the 8 V typically needed by these controllers. This problem was overcome by using Lithium Ion (Li-Ion) cells. While these offered nearly the same capacity and power to weight ratio as Li-Po, Li-Ion was a safer chemistry and so was not limited by the rules. A three cell Li-Ion battery would provide a minimum of 9 V. The downside was the reduced charging rate which could struggle with prolonged regenerative braking, and the need to construct a battery pack.

The controller chosen was rated for 20 A continuous, up to 30 A peak, with a minimum supply voltage of 9 V. The controller had the ability to limit the effective maximum voltage at the motor to prevent damage as the motor was only rated for 8.4 V. It used trapezoidal commutation and could be directly interfaced with the hall effect sensors on the motor. The controller also offered acceleration and deceleration limits as well as PID speed, torque and position control. It could be interfaced with a microcontroller over RS232 to both be controlled and to relay measurements taken of the motor.

4) Battery pack

As previously explained, Li-Ion cells were chosen to circumvent competition rules on voltage limits, while maintaining the superior power to weight ratio of Lithium chemistries. Capacity was less of a concern given the short length of the track, however, the requirements were still calculated.

Using the maximum torque calculated in (11), the maximum current (during maximum acceleration) could be found from (15) and was found to be 14.6 A. A worst-case current draw scenario would come from a track where straight sections were evenly divided and separated between 90˚ corner pieces. The corner pieces would slow the car to a constant speed each time they were encountered, so by evenly distributing the straights the acceleration distance is unchanged but the average speed, hence time spent accelerating, is maximised. The use of 90˚ rather than 180˚ pieces ensures greater distribution of straight sections.

As previously explained (VI.A.2), a centre line reduces the average speed more than a racing line, and so a car following a centre line would spend the most time navigating the track. Secondly, a centre line does not protrude into the straights as a racing line would. This both maximises the straight distance over which the car can accelerate, further increasing the current used and simplifies the distance calculations Therefore, the scenario assumed centre line navigation.

It was assumed that the car would be accelerating maximally up until the middle of all straight sections, at which point it would brake maximally. During corners the car would maintain its speed and no current would flow. Further, it was assumed that deceleration would be achieved by plugging and use the same current as acceleration. Although in reality regenerative braking would recover some of the energy used in acceleration, the scenario assumes no losses during corners or elsewhere, therefore ignoring regeneration should increase margin for error.

The example track [8] was used and simplified to assume the length of a chicane and crossing point were each the same as a straight section. This resulted in 16 straights and 16 90˚ bends. Therefore, each straight would be the length of a single straight track piece. The maximum speeds on the corners and straights were calculated using (1) through to (9) and then used to calculate average time spent on each type of track piece. The cumulative time spent on the straights along with maximum current draw was used to calculate the battery capacity requirement of 856 J per lap. The intermediate values can be seen in TABLE II.

TABLE II
ENERGY USE PER LAP (ANDREW)

Parameter	Description	Value	Unit
l	Length of straight	0.690	m
s	Displacement to centre of straight	0.345	m
v_c	Velocity through corner	1.98	ms^{-1}
a	Traction limited acceleration	8.82	ms^{-2}
v_{max}	Maximum velocity on straight	3.16	ms^{-1}
v_{mean}	Mean velocity on straight	1.58	ms^{-1}
s_{total}	Total displacement of straight sections	11.0	m
t_{total}	Total time spent on straight sections	6.98	s
I_{max}	Current at maximum acceleration	14.6	A
V_{mot}	Motor voltage	8.4	V
E_{lap}	Energy use per lap	856	J

Approximate energy usage per lap assuming no current drawn through corners, maximum acceleration on straights and braking by plugging. Example track arranged into individual straight sections separated by 90˚ corners.

The energy use was roughly two orders of magnitude lower than that stored in a typical Li-Ion cell and so the capacity was not a concern, even with the crude estimations made. Power density was of greater concern since the regenerative braking current would have been similar in magnitude to the 15 A acceleration current. The cell had to be specially chosen to maximise the charging current. The one chosen was capable of supplying 30 A continuously, however it was still only rated at 6 A continuous charge. No peak charge current was specified. It was decided that the brief charge periods and the short time taken to complete a lap would not be considered continuous charging, and so it would be safe to use the cells so long as their temperature was frequently checked.

A three series, one parallel, battery pack was made using the chosen cells. This would form a 2500 mAh battery, with a 9 V to 12.6 V range, 30 A continuous discharge current and 6 A

continuous charge current. Assuming the motor controller was 100% efficient and stepped down the 10.8 V nominal battery voltage to the motor's maximum of 8.4 V, then each lap would require 22 mAh from the battery. At this rate the battery could provide in the order of 100 laps.

The pack was made from individual cell holders and mounted to the base-plate of the chassis alongside the motor to lower the centre of gravity. This also allowed the cells to be removed for charging, avoiding the need for cell balancing and charging circuitry on board the car. A Lithium cell monitor alarm was fitted to alert the user if any cell voltage was too low. An inline DC car fuse rated at 30 A was installed to protect from short circuit conditions.

5) Electronics mounting board and camera mount

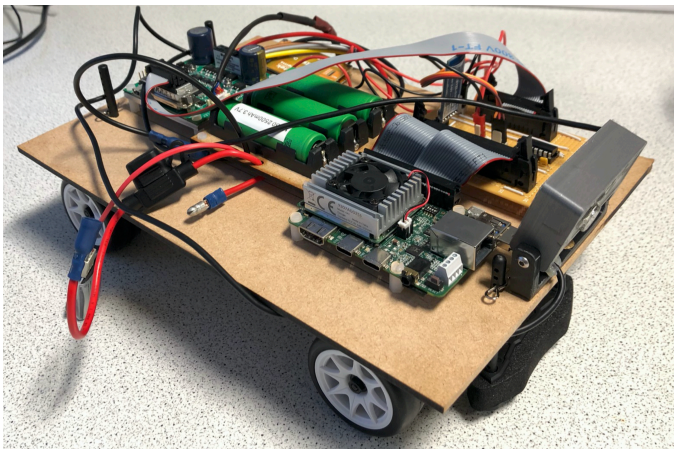

Fig. 12. Prototype car design from December 2019. Changes from this point were: the replacement of the camera with the new Pixy2 camera and mount, replacement of the Coral Dev Board with a FRDM-K64Z, and mounting the battery pack to the bottom plate of the chassis.

A prototype board to mount electronics to the car was manufactured. The board was made from MDF for easy modification while the electronics were being finalised (Fig. 12). A camera mount (not shown) for the Pixy2 was also manufactured. The mount allowed for height and angle adjustment and simple fixing to the MDF board. The crucial cylindrical fixing which allowed the camera tilt to be adjusted was designed to be removed from the prototype camera pole and reused in a final fixing.

B. Microprocessors and communication

A significant part of the project involved selecting appropriate computing hardware on which to run the computer vision and control algorithms and ensuring all parts could communicate.

1) Coral dev board (Hamish)

The NXP rules state that all processing boards must be NXP chipsets, to be able to read and process from a camera in real time whilst dealing with motor signalling and thus a powerful board is needed. The one Coral dev board is an NXP board designed for Neural network processing including an onboard tensor processing unit. The Coral dev board also runs a specialised Linux OS called Mendel, designed to interface with the specific hardware and allowing for USB devices to be connected simply. Setting up this board was much more difficult than expected given a complex flashing procedure and a badly timed MacOS update breaking the communication protocol. The coral dev-board has inbuilt PWM, I^2C, SPI and UART controllers for easy communication with all desired devices. Unfortunately, the Coral dev board designers defined many of the systems with only neural networks in mind making simple IO and low level control much harder to access given an old file pin access style to accommodate for the multi-user OS architecture used. This OS architecture designed for single process processing meant context switching for any IO or camera reading which is constant in this system.

2) Freescale board (Hamish)

To get past the issues caused by the running of an OS on the main controller board and slow connections, the Coral dev board was replaced by an embedded controller. Due to the rules of the NXP cup, the processing board must be an NXP device and the availability of certain Freescale (owned by NXP) boards, the K64F was chosen due to the higher processing power and more modern design. This board has inbuilt I^2C, SPI and serial UART controllers to allow for easy communication with all devices. The K64F board uses a 32-bit ARM-based processor allowing for programming using ARM MBED studio and C/C++ programming control.

An older version of the K64F is the KL25Z with a much slower clock rate and smaller memory but more integrated I^2C and serial UART connections for more separate devices to be ran in parallel. This extra UART channel became necessary for programming the device as the car is programmed over Bluetooth for safety, allowing for a wireless shutdown. This meant the less powerful but cheaper, smaller and lighter board was used. This lack of processing is acceptable given the moving of computer vision from the master processing board to the Pixy2 slave.

3) Data communication (Hamish)

To communicate with all of the cars peripherals many different communication techniques are used based on what the slave controller requires. The servo uses 50Hz Pulse Width Modulation (PWM) to control the position of the servo using an internal PID controller. The KL25Z and K64F both have many analogue output pins with variable frequency PWM generators. The Pixy2 accepts many communication protcols, specifically SPI, I^2C, serial UART and USB for Linux based controllers. For a non-Linux based system, SPI is recommended for the highest data rate with the KL25Z and K64F each having a single hardware SPI interface. For any extra devices to be connected over SPI slave select pins could be used with slight modifications in the library given a current lack of implementation otherwise SPI can be manually programmed on any IO pins and used as another interface. Both the motor controller and Bluetooth wireless programming module HC05 communicate via UART, unfortunately the K64F documentation doesn't clearly state the UART positions except from a single TX/RX connection, with some extra testing the K64F UART pins may be found and used but naturally the K25Z is more well documented.

4) Breakout Shield PCB (Andrew)

A PCB was designed to conveniently access the ports on the microcontroller as well as contain various level shifting circuits for interfacing. Using an open source footprint for the microcontroller, the PCB was designed to fit directly on top. The PCB was intended to replace the matrix-board based breakout-board made previously. The schematic can be found in Appendix Fig. 18.

A level shifter was required to interface the PWM signal between the high voltage servo logic (7.2 V) and the 3.3 V level logic of the microcontroller. A line driver IC and accompanying support capacitors were used to interface between the 3.3 V level UART of the microcontroller and the ±15 V RS232 used by the motor controller.

A port was added to allow a removable serial Bluetooth module to be connected to the UART of the microcontroller. This would allow for simpler debugging and control during testing but could be removed to qualify for the competition. Additional ports, which were not on the original prototype board, were added to connect the laser and sonar modules to the microcontroller. A MOSFET was used to allow brightness adjustment of the lasers using PWM.

The PCB design was not intended to be a primary objective but was completed after the early termination of the project.

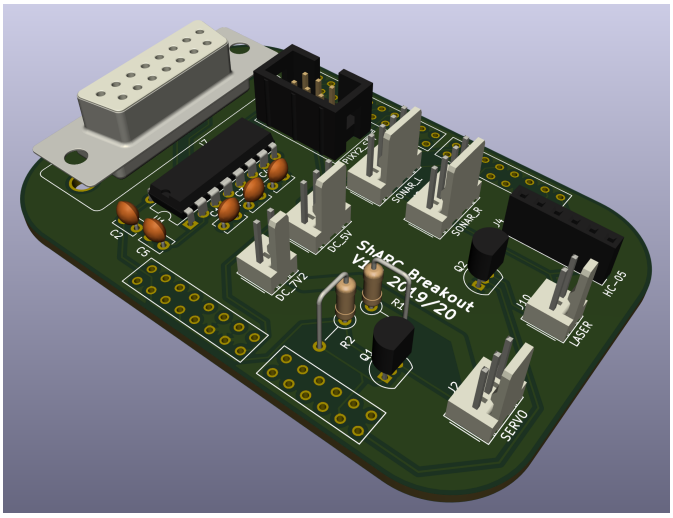

Fig. 13. 3D render of breakout shield for FRDM-K64F microcontroller. The underside of the shield would be populated with pins and mounted directly on top of the microcontroller.

5) Speed controller setup and interfacing (Andrew)

Using the RS232 link made in the breakout board, the microcontroller was interfaced with the speed controller. The speed controller was set up to run a watchdog timer which would emergency stop the motor if not reset by the microcontroller every 500 ms. This was to prevent a run-away car in the event of a communication or software fault. A C++ class was written to initialise the communication port, reset the watchdog and allow a speed to be relayed from a Bluetooth connected computer to the motor controller. The remote control feature was to simplify testing of steering control and was due to be replaced by an autonomous speed setpoint algorithm.

The PID within the motor controller was only roughly tuned. Proportional and derivative control were deactivated and the integral gain was set to one. This was the recommended starting point from the manufacturer [63]. This gave acceptable performance at all but low speeds. At low speeds oscillation was observed. The controller would require further tuning particularly once a variable speed setpoint algorithm is developed. Tuning was not completed due to the early termination of the project.

6) Pixy2 Camera library porting (Andrew)

The computer vision camera used for navigation came with an open source communications library written in C++ for the Arduino platform. Due to various Arduino specific functions used in the library, it was not compatible with the Arm Mbed OS platform running on the FRDM-K64F microcontroller controlling the car. Porting involved re-writing these functions, mainly print and timing based, to use Mbed compatible alternatives. The ported library was fully functional and supported all features of the original.

C. Addon modules

1) Lidar obstacle detection (Andrew)

One potential method for detecting the presence of an object was to project a laser beam in front of the car and use object detection methods to detect the coordinates of the reflected laser point. So long as the laser beam did not originate from the camera lens, the interception of a beam by an object would cause movement in the reflected point. Assuming the laser beam was centred and parallel with the direction of the camera, any object with a non-zero height would cause the point to rise vertically. The object detection algorithm in the Pixy2 camera would be able to track the point and a rise in the Y coordinate could trigger obstacle avoidance behaviour from the car. By calibrating the known position of the laser point when no object is present, the controller could measure the displacement of an object in front of the car. Multiple points could be projected to give greater resolution over the horizontal position of the object. This would be possible with the Pixy2 which can track multiple objects.

The Pixy2 camera used image hue to detect objects. Therefore, it was crucial that the laser point would not be too bright which could cause over-exposure and make the point appear white. Conversely, if the point was too dark then it could be lost against the ambient light. Tracking would be lost in both of these instances. Although brightness could be controlled to an extent by altering the camera's exposure settings, having the additional option of altering the laser brightness would be beneficial.

It was important that safety was maintained with the use of lasers. Although no rules had been set by the competition, it was decided that the maximum class should be a 3A (< 5 mW). These pose a low risk of eye damage from accidental exposure and no risk from diffuse radiation caused by the reflected beam scattering off an object.

A method was required to produce multiple beams to detect horizontal object position. Solutions such as beam splitters use partial reflection to separate the beam but require precise alignment and are bulky. Rotating mirrors use a pulsed beam to give the illusion of splitting, however this could result in

interference with the camera shutter rate. These are also expensive, bulkier and require even more precise alignment. Diffraction gratings are small and inexpensive but cause variation in beam brightness, which would cause exposure problems. Transmission grating beam splitters offer the same practical benefits as diffraction gratings, without brightness problems, but are prohibitively expensive. In addition, all these solutions apart from the rotating mirror would require a more powerful and dangerous laser for there to be sufficient brightness left in the split beams. As a result, the least expensive and simplest method was to use multiple laser diodes. The diodes were available within brass heat sink housing and were exceptionally small and inexpensive. Five diodes were purchased which would give a 1 m detection width for tracking the 0.2 m wide obstacle.

It was later found that the Pixy2 camera was incapable of simultaneous object detection and line detection (for navigation). Without purchasing a second camera, a solution would have been to rapidly switch between detection algorithms. Since the object detection lap could be run at any speed, high precision line following was not necessary. An experiment discovered that the switching time from line tracking to generating the first object detection output took 4 ms and the reverse process took 50 ms. An approximate 10 Hz refresh rate for each type of detection may have been acceptable for a low speed run.

Although the lasers were received, there was insufficient time to build a mount or test the system due to the early termination of the project.

2) Sonar obstacle detection (David)

As previously mentioned, due to the limitations of only being able to run one detection algorithm at a time, the decision was made to develop a solution for the object avoidance and emergency brake which only used the ultrasonic sensors. Development of the laser detection could then continue with the guarantee that the ranging system could work independent of a laser module, in the event of finding its implementation not feasible.

HC-SR04 ultrasonic sensors were purchased due to their low price, with a per unit cost of around £2, extensive documentation, and simplicity of design. To measure distance using the sensor, an ultrasonic library was written to use on the IDE 'mbed studio' with the following key functions:

Sonicsensor() – instantiates an ultrasonic sensor using pins and parameters (measure rate, distance threshold etc.) inputted as the arguments for this function.
startTicker() – initiates measurements, sending a trigger pulse at the measure rate set by Sonicsensor().
startEcho()/endEcho() – starts and stops a timer to measure the pulse length of the return echo pulse.
distCheck() – calculates and returns the distance using the time measured from the echo function.

To perform these key functions, hardware interrupts were needed to ensure measurements were regular and the echo pulse was timed accurately. As a measurement is started via a pulse sent from the trigger pin, the mbed ticker interface was used, which creates a recurring interrupt that executes a defined function each time. By creating a ticker which sets the trigger pin high at the refresh rate, followed by a second nested ticker that cut the pulse 10µs after the initial trigger (the trigger input pulse width), the result would a regular trigger pulse with consistent timing. Timing of the return echo was then achieved via two hardware interrupts that started and stopped the timer on the rising and falling edges of the signal. Finally, the time is divided by the sonar wave's time taken per cm (doubled to include return journey) to calculate the distance from the transducer in cm.

The emergency brake challenge is thus performed by setting a threshold distance for the sensor, in which the command to brake the car was triggered upon going below this value - this had been written but was not tested due to COVID-19 restrictions.

The object avoidance system was built by expanding and modifying the emergency brake program, with the instantiation of two sonar sensors as left and right – while the HC-SR04 had a 30 degrees measuring angle, reducing the positional accuracy of the device, the approximate distance would be sufficient to determine which side the object would be on and thus which side of the track to travel around. After the distance threshold is flagged, object's location would be found by determining which sensor had the smallest reading. The pixy2's line detection algorithm (discussed in section D) would be switched to black line detection from white to output two vectors for the left and right track edges. Identifying which vector was for each edge was done via comparison of the x-coordinates of the vectors' bases, and once done the steering control could be set to follow the edge opposite to the object. This program was largely written but was left incomplete due to COVID-19.

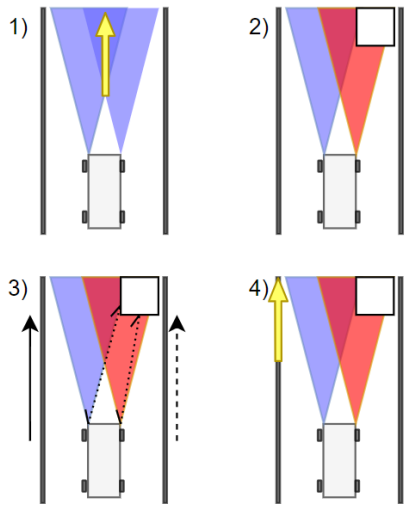

Fig. 14: Car model showing 1) centreline program, 2) detection of object via sensor readings below threshold, 3) determination of object location via comparison of sensor readings, and selecting new vector to follow, 4) reinitiating new centreline program

D. Computer vision and steering

Computer vision is, by definition of the rules, required to be the primary source of track data and therefore of high importance to the success of the project and therefore should be a main focus of the project.

1) Hough transform (Hamish)

The initial computer vision system used a Hough transform from the OpenCV Python library to detect the track edge lines and create a determinable centre line equation as shown in Fig. 15. Whilst this method worked well for determining straight lines and being able to calculate line equations, the limitations of a Hough transform mean this system is impossible to implement for detecting corners without massive re-designs. The Hough transform would also struggle with intersections due to the large broken lines; this could be partially fixed by going straight with no detected lines but this is far from a perfect solution. This method would also struggle with the chicane track piece as intermittent lines may be detected and put the algorithm off course.

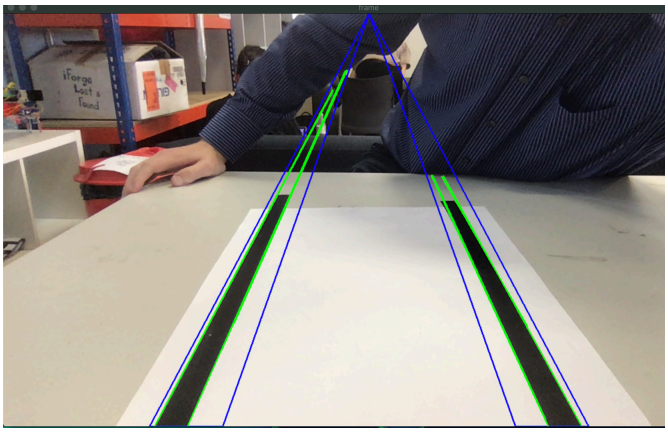

Fig. 15. A Hough transform implementation with detected lines in green. The blue lines represent a masked region to remove noise caused from working in non-test conditions.

2) Black averaging (Hamish)

Taking advantage of the simplistic track is key, to get around the cornering and chicane issues found from the Hough transform, a more analogous approach must be taken. A common line-following robot computer vision technique is black averaging where the horizontal position of black pixels on the screen are averaged to create a single desired line. As instead this system uses two lines, two separate line averages could be used to track these lines as seen in Fig. 16.

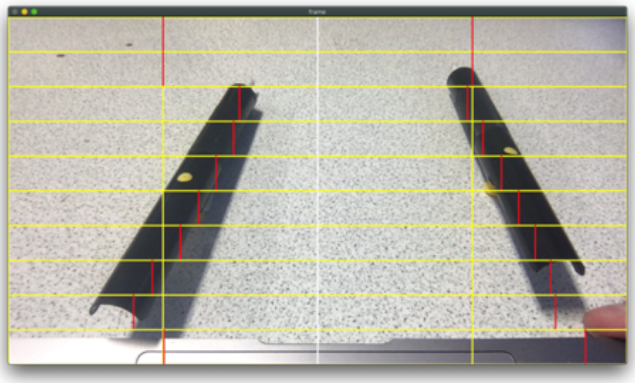

Fig. 16. Dual Black-Averaging line detection example. Here the white line separates left track from the right track, yellow lines segment the display into averaging regions for computation efficiency and the red lines show the average detected black within the regions

This system is programmed using OpenCV to read the image and convert to a black and white image, blurred by a 5x5 gaussian blur kernel to remove noise followed by a binary linear thresholding to binarily classify a pixel to image or track. This feature map is then averaged in sections as seen in Fig. 16. Dual Black-Averaging line detection example. Here the white line separates left track from the right track, yellow lines segment the display into averaging regions for computation efficiency and the red lines show the average detected black within the regions, to generate the averaged track lines. This algorithm ran on the Coral dev board running Linux discussed previously. As this camera has its own onboard processor, communicated over USB, the systems input lagged a few hundred milliseconds behind which is not acceptable for a real time processing device. If the camera could be replaced and connected via a high speed, low latency connection this type of processing may work well. The issue may also be enhanced due to the system call method of reading and writing to IO creating large comparative delays to a single user space device and thus an embedded computer vision method is advised.

3) Neural networks (Hamish)

As the Coral dev board focuses on neural network methods given the tensor processing unit and the recent advancements in neural networks, neural networks became a recurring conversation. Using physics, car and track simulations, a neural network could be trained for days to find the best method of not only steering but viewing the track and driving speed to complete the track as fast as possible. Such simulations are difficult to create to an acceptable standard of physics and vision accuracy and such for a project with such wide scope is likely too much work. Instead a convolutional neural network could be trained using human data. This could work by driving the car by hand around a track storing the input data such as camera feed, current speed and current steering angle as well as output control data such as desired steering angle and desired speed. A simple convolutional neural network can be used to generate and discriminate the data to mimic the human behaviour; this would likely be very accurate due to simple input and output data and the output efficiency would be mainly dependent on the competency of the human driver trained against. This method of driving was never implemented but a proof of concept was created using Google Collab.

4) Pixy2(Hamish)

The NXP cup organisers compiled a list of recommended hardware such as their kits which were decided against with better results. The computer vision method recommended was the use of the Pixy2, after using the aforementioned techniques, the Pixy2 was bought to use instead allowing for an embedded system for computer vision compared to an operating system. The Pixy2 has many inbuilt features such as line tracking, barcode detection and colour/object detection. The Pixy2 still requires programming to be controlled and can be done so over I^2C and SPI, SPI was chosen for the faster data rate. The Pixy2 could be entirely controlled by an NXP processing board and tweaked using desktop software allowing for full embedded control of the device whilst having the benefits of partial GUI development for tweaking values such as thresholds. The Pixy2

had many modes, each allowing for the ability to complete each of the required NXP races. Unfortunately, these modes are mutually exclusive and require rapid switching to complete the desired functions.

An official Pixy2 library had been made to implement the system using Arduino but due to the rules these Atmel processors are not allowed and instead an NXP chipset must be used. To run the code the library was re-written replacing any Arduino command references to the C++ equivalent code. This library was used to program the car to detect the track vector, this was done by detecting a white line on a black background. By doing this the car automatically detects the vector of the centre of the track rather than processing the outer vectors generated by detecting the black lines. This single vector was then processed and sent via the SPI interface to the microcontroller for processing.

5) Steering Control (David)

Improving upon the original black averaging steering system, a PID controller was implemented to use the Pixy2 camera and its line tracking system. This line tracking program could recognise a black or white line and output a vector indicating its direction, which was used to follow the centre of the white track. In the control system diagram shown in Fig. 17, the aim of the system is to angle the car to consistently follow the centre of the track - to monitor this, the measured process variable, the angle of the vector found by the Pixy, would be compared against a zero degrees setpoint to give an angular error, which was then fed into the PID controller. The error is multiplied, integrated and differentiated, and combined with the respective proportional, integral and differential gains, and all three components are summed to give a single controller output. This output is scaled to a PWM signal and output to the servo motor by the microcontroller, where the servo could rotate the arm to the desired angle for the wheels. Forward motion of the car would result in a new car angle, which would then be measured by the pixy and fed back into the system. This PID control had been completed before COVID-19 restrictions but was only initially tuned and tested.

Additional steps are required for certain challenges, notably the object avoidance challenge, which require further modifications to the system to perform correctly. As mentioned previously, the final stage of object avoidance follows a track edge instead of the white centre to pass the object. The vector angle feedback must be modified at this point by switching the Pixy2's line algorithm mode from white to black detection and using the correct vector to calculate error.

VII. Discussion (Andrew)

Due to the early termination of the project and cancellation of the competition resulting from COVID-19, none of the project aims were met. Although this also affected how many of the objectives could be met, many were close to completion.

The objective to develop a car was completed. A fully functional car for use as a test platform was developed earlier on in the project. The drivetrain parts were chosen to fully utilise the capability of the chassis while meeting all competition rules. A BLDC motor and controller were paired with a Li-Ion battery to improve the system power to weight ratio and ultimately improve performance. This differs from the majority of line following robots used in research which relied on BDC [10] [11] [12]. The chassis was chosen to maximise traction while being small enough to fit on the track and allow for simple modification to fit the electronics on. It also utilised Ackermann steering, further differentiating it from many line following robots in research [10] [11]. The use of 4WD and a single motor with a differential separated this chassis from even those which used Ackermann steering [12]. Although the performance of the car was expected to meet requirements, there was no way to test this as the control testing side of the project was cut short by early termination of the project. The car was also heavy as the electronics plate and camera mount had not been finalised and were still in their prototype forms.

The objective to develop computer vision based steering control was very close to completion. The Pixy2 camera was used as the primary computer vision device and had been successfully interfaced with the microcontroller to develop a closed loop PID steering controller. The use of a continuous PID controller improves on previous work with the Pixy2 which used discrete thresholds of angle error to issue fixed steering angles [10]. At the time of project termination, the PID steering controller was being tested in situ with the car on a short section of example track. One of the problems noted was the narrow camera angle causing the Pixy2 to lose line tracking around corners for example. A possible solution to this was to mount the camera to a servo controlled by a separate PID controller which attempts to maintain the Pixy2 output vector in the centre of the frame. This system would ensure the camera

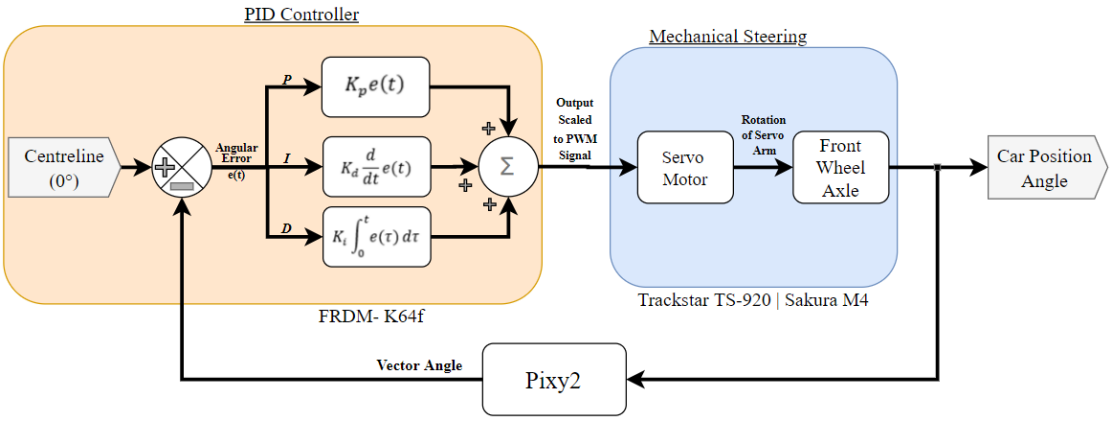

Fig. 17: System diagram of the steering control, showing the actions performed by the microcontroller (yellow), response of the motor (blue) and method of feedback

was always pointing down the track regardless of the movement of the car. The known angle of the camera servo alongside the Pixy2's vector could be used to calculate a true steering error taking into account the additional rotation. However, embarking on developing such a system would have required more testing of the original steering system to decide if the added complexity would be worthwhile.

A racing line algorithm was relegated to a secondary objective in the interim report due to time constraints. However, this assumed that the computer vision system would be on OpenCV. The Pixy2 is much more limited in terms of the outputs it can generate, for example, the radii of curves cannot be measured. This has prevented the development of a racing line algorithm which has instead been absorbed into tuning of the steering and speed controllers. Although the Pixy2 was a more limited platform than OpenCV, the decision to use it over was justified to establish a working prototype by competition deadlines.

The emergency braking objective was very close to completion. At the time of project termination, the sonar modules used to detect a solid object in front of the car were interfaced with the microcontroller and correctly reporting distance. Functioning code was also written to send speed commands to the motor controller. The only remaining work was to physically mount the sonar modules and set a threshold distance which would send a zero speed command to the motors. Although no testing had been done, it is believed that the system would be reliable as there was no reliance on computer vision and training the system to detect an object. A computer vision approach would have been complicated by the Pixy2 camera's inability to simultaneously run line detection and object detection. If OpenCV had been used then a computer vision detection system would have saved some weight and physical complexity by removing the sonar. However, the reliability of sonar would likely have outweighed these minor benefits.

The obstacle avoidance objective was in progress at the time of project termination. The primary sonar based method was capable of measuring the distance to objects. Although the algorithm to then determine the object position and navigate around it had not been implemented, there was a plan of how this could be done. Had it been fully implemented, the sonar system may have suffered from low resolution as only two sensors were used. The alternative laser based lidar system would have provided higher resolution given the five points projected. However, due to the Pixy2 being unable to simultaneously run object and line detection algorithms steering and detection performance may have deteriorated with this system. If necessary, a secondary Pixy2 camera could have been purchased, however an OpenCV approach would have been the ultimate solution combining object and line detection if the additional resolution of a lidar system was worth the complexity over the sonar system.

Development of speed limit detection had not begun by the end of the project. The other objectives were prioritised over this as the Pixy2 camera had built-in barcode detection as part of the line following algorithm. This should have made teaching and recognising the speed limit barcodes a quick process. This was one of the areas in which the Pixy2 camera should have accelerated implementation over an OpenCV approach.

VIII. Conclusions and Recommendations (David)

In conclusion, while the aims of the project were unable to be fully completed due to COVID-19, as the first year competing in the NXP Cup, our work presents a strong base for future teams to continue. Many key objectives such as the car build have been completed, and what has not, such as the ranging system and PID controller, has had significant progress made.

To iterate upon this year's design, there are several improvements that could be implemented that could yield significant improvements to the final performance of the car, of note being items that were removed from our aims and objectives due to time constraints. The implementation of a cornering algorithm over the current centre line method would maximise the strengths of the motor, as well as optimising additional challenges such as figure of eight with a dedicated mode of operation as opposed to using the main race algorithm, which could potentially speed up performance of the car across the overlap. Lastly, while the speed controller is fully developed, the speed control would vary speed proportional to the error - developing a new system that could further vary speed based on upcoming sections of track, combined with a cornering algorithm would make significant improvements in the race performance.

IX. Future Work Prevented by Covid-19 (Hamish)

Due to the Covid-19 pandemic, certain parts of the project couldn't be implemented in time as explained here. Unfortunately, due to the travel and gathering restrictions placed nationally the final competition couldn't safely run meaning the project, even if completed, would never compete in the 2020 competition.

1) PID Tuning

The final change made before restrictions were placed around university access to laboratories was the implementation of a steering PID controller to dampen steering oscillations. Once this was implemented, it was tuned by placing the car on the track and increasing and decreasing the PID values based on how well the car navigated the track. The next step of tuning was to use a tuning method, specifically Ziegler Nichols tuning to have a well tuned car ready to navigate the track.

The motor controller board used also had a software PID controller built in, this comes with defaults for the motor but with extra time tuning this PID could've been an easy improvement to the cars performance. To drive this PID the car needed to detect the best speed for the current manoeuvre compared to the constant speed methodology. This desired speed would likely be as simple as an inversely proportional value based on the car steering angle to reduce speed in turning and increase in the straights.

2) Top Plate

To improve the design of the car, a top plate was going to be designed to replace the MDF board holding all components to the car. This was designed to make the project more professional, tidy and cables to make sure they didn't drag or get caught in gears, make the car lighter and more aerodynamic and allow for easy switching of components on and off the car for separate races. However for the top board to be designed, component placement needed to be finalised including position and wiring of the laser and transducer modules. The materials and facilities to laser cut acrylic once the design was made were available for printing at any time in the university facilities. With time to finalise the placement of these modules and create a 2D measured model this top plate could've been implemented.

3) Laser obstacle avoidance

The laser system designed to create a visual indication of a white object, made to complete the obstacle avoidance test, was partially completed but couldn't be completed due to the pandemic. The lasers for detection were purchased and tested ready for mounting and driving, the final method of mounting wasn't finalised but was made to be modularly attached for power allowing to be taken off for all other races. The voltage to power the laser was completed by nature given the internal 5V line generated for powering the processing board. The software for the laser module depended on the built in object/colour detection of the Pixy2 that could be setup to detect specifically small single colour objects and return the position. After setting this up the code would need to detect where the box is and then drive to the side possible by changing the desired vector for the section of the course. Upon completion of the mounting and software, this laser obstacle avoidance would be ready for the NXP cup.

4) Emergency brake transducer

The emergency brake system, used to detect a wall on the track for the emergency stop race, used an ultrasonic transducer module to read distance ahead of the car. The ultrasonic sensor, like the laser system had been bought and tested but also the program written to control the HR-SR04 device, waiting only for a detachable mounting solution before being race-ready. This device has no requirement for change of the Pixy2 computer vision and is a simple addition for easy points in the race.

5) Speed limit detection

The speed limit detection, designed to detect either three or four vertical lines on the track, was planned to use the internal bar-code detection algorithm by training with new specific barcodes. Instead of this the internal colour code classification could be used giving a similar labelled output. The barcode detection system was un-developed and un-tested but once setup no changes would be needed in terms of mounting or infrastructure given the Pixy2 basis. After classifying the barcode, the system would update the speed of the car depending on a slow down or speed up command. This change would be minimal given the current design of the control system. A similar test can be seen on the Pixy2 website using the barcode detection and line following together with the respective code likely making this easier than developing from scratch. The implementation of this speed limit sign detection would allow completion of the speed limit zone track race.

6) Testing

After each of these modules were completed, time was allocated to test each of the systems and switching between races as efficiently as possible to classify the completion against aims and objectives and allow for any modifications to be made. Unfortunately given the pandemic, all systems weren't tested to the degree necessary for classification. To test the systems each race would've been emulated using the track pieces available tuning values and making sure the car can be changed to work for different races without re-programming the car as specified in the NXP cup rules. This testing would also allow for much more decisiveness over the efficiency of the methods used for each of the NXP cup races especially the more complex races.

X. References

[1] R. E. Fenton, "IVHS/AHS: driving into the future," *IEEE Control Systems Magazine1,* vol. 14, no. 6, pp. 13-20, 1994.

[2] V. G. a. K. Kuhnerf, "Towards A Vision Based Robot With A Driver's License," in *IEEE International Workshop on Intelligent Robots*, Tokyo, 1988.

[3] S. Tsugawa, "Vision-based vehicles in Japan: the machine vision systems and driving control systems," in *Budapest: IEEE International Symposium on Industrial Electronics Conference Proceedings*, Budapest, 1993.

[4] S. S. e. al., "Vision-based road following in the autonomous land vehicle," in *26th IEEE Conference on Decision and Control*, Los Angeles, 1987.

[5] O. A. A. Ollero, "Predictive path tracking of mobile robots: Application to the CMU - NavLab," in *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, Pisa, 1991.

[6] T. J. D. Pomerleau, "Rapidly adapting machine vision for automated vehicle steering," *IEEE Expert,* vol. 11, no. 2, pp. 19-27, 1996.

[7] S. Thrun, "Toward Robotic Cars," *Communications of the ACM,* vol. 53, no. 4, pp. 99-106, 2010.

[8] NXP, "The NXP Cup EMEA," [Online]. Available: https://community.nxp.com/servlet/JiveServlet/download/101612-7-431006/NXP+CUP_track_configurations_2018_19.pdf. [Accessed 9 May 2020].

[9] NXP, "The NXP Cup Official Rules," 27 Spetember 2019. [Online]. Available: https://community.nxp.com/docs/DOC-335269. [Accessed 12 May 2020].

[10] J.-H. Li, Y.-S. Ho and J.-J. Huang, "Line Tracking Pixy Cameras on a Wheeled Robot Prototype," in *2018 IEEE International Conference on Consumer Electronics-Taiwan*, Taichung, 2018.

[11] A. M. E. Abdelkafi and S. Wang, "Uncalibrated differential drive visual line follower automatic vehicle," in *CSAA/IET International Conference on Aircraft Utility Systems*, Guiyang, 2018.

[12] J. Hrbáček, T. Ripel and J. Krejsa, "Ackermann mobile robot chassis with independent rear wheel drives," in *Proceedings of 14th International Power Electronics and Motion Control Conference*, Ohrid, 2010.

[13] A. H. Ismail, H. R. Ramli and M. H. Marhaban, "Vision-based system for line following mobile robot," *10.1109/ISIEA.2009.5356366,* vol. 2, pp. 642 - 645, 2009.

[14] J. Sarwade, S. Shetty, A. Bhavsar, M. Mergu and A. Talekar, "Line Following Robot Using Image Processing," in *2019 3rd International*

Conference on Computing Methodologies and Communication (ICCMC), 2019, pp. 1174-1179.

[15] B. Åstrand and A.-J. Baerveldt, "A vision based row-following system for agriculture field machinery," *Mechatronics,* vol. 15, no. 2, pp. 251-269, 2005.

[16] S. Ishikawa, H. Kuwamoto and S. Ozawa, "Visual navigation of an autonomous vehicle using white line recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 10, no. 5, pp. 743-749, 1998.

[17] . I. Akkaya, . M. Andrychowicz, . M. Chociej, . M. Litwin, B. McGrew, . A. Petron, . A. Paino, . M. Plappert, . G. Powell, . R. Ribas, . J. Schneider, . N. Tezak, . J. Tworek, . P. Welinder, . L. Weng, . Q. Yuan, W. Zaremba and . L. Zhang, "Solving Rubik's Cube with a Robot Hand," vol. arXiv preprint arXiv:1910.07113, 2019.

[18] A. A. G. G. a. P. M. S. Dominguez, "Comparison of lateral controllers for autonomous vehicle: Experimental results," in *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, 2016.

[19] W. R. R. e. al., "First results in robot road-following," in *International Joint Conference on Artificial Intelligence* , Morgan, 1985.

[20] Y. S. L. C. K. H. a. D. C. Y. Chen, "Optimization of Pure Pursuit Controller based on PID Controller and Low-pass Filter," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, 2018.

[21] S. L. W. H. Myungwook Park, "Development of Steering Control System for Autonomous Vehicles Using Geometry-Based Path Tracking Algorithm," *ERTI Journal,* vol. 37, no. 3, pp. 617-625, 2015.

[22] H. B. Stefano Stramigioli, Geometry and Screw Theory for Robotics, 2001.

[23] J. S. Wit, "PHD Thesis: Vector Pursuit Path Tracking for Autonomous Ground Vehicles," University of Florida, Florida, 2000.

[24] J. B. F. M. P. M. B. Thuilot, "Accurate Automatic Guidance of an Urban Electric Vehicle Relying on a Kinematic GPS Sensonr," in *5th IF ACIEURON Symposium on Intelligent Autonomous Vehicles* , Lisboa, 2004.

[25] C. L. F. L. C. a. W. L. Ma, "Path Following Based on Model Predictive Control for Automatic Parking System," SAE International, 2017.

[26] P. Z. e. al., "Design of a Control System for an Autonomous Vehicle Based on Adaptive-PID Controller," *International Journal of Advanced Robotic Systems,* Vols. 9, , no. 4, 2012.

[27] J. C. G. N. R. Kapania, "Design of a Feedback-feedforward Steering Controller For Accurate Path Tracking and Stability and the Limits of Handling," *Vehicle System Dynamics,* vol. 53, no. 12, pp. 1687-1707, 2015.

[28] P. Yedamale, "Microchip," 2003. [Online]. Available: http://ww1.microchip.com/downloads/en/appnotes/00885a.pdf. [Accessed 8 May 2020].

[29] D. Lin, P. Zhou and Z. J. Cendes, "In-Depth Study of the Torque Constant for Permanent-Magnet Machines," *IEEE Transactions on Magnetics,* vol. 45, no. 12, pp. 5383-5387, 2009.

[30] J. R. Hendershot and T. J. E. Miller, Design of Brushless Permanent Magnet Motors, Oxford: Magna Physics/Clarendon, 1994.

[31] Y. Huang, Y. Xin and W. Zhang, "An improved BEMF detection method for sensorless BLDC motors," in *2008 IEEE International Conference on Industrial Technology*, Chengdu, 2008.

[32] M. Rakesh and P. V. R. L. Narasimham, "Different Braking Techniques Employed to a Brushless DC Motor Drive used in Locomotives," *International Electrical Engineering Journal ,* vol. 3, no. 2, pp. 784-790, 2012.

[33] O. Mayr, The Origins of Feedback Control, Cambridge: MIT, 1970.

[34] E. L. Owen, "Origins of the Servo-Motor," *IEEE Industry Applications Magazine,* vol. 2, no. 2, pp. 74-, 1996.

[35] T. L. Z. Z. S. W. a. H. G. Q. Lin, "Application of the Magnetic Encoder in Actuator Servo System," in *2007 International Conference on Mechatronics and Automation*, Harbin, 2007.

[36] Baldor Electric Company, Servo Control Facts: A Handbook Explaining The Basics of Motion, Fort Smith, Arkansas: Baldor Electric Company, 2013.

[37] H. Zhou, "DC Servo Motor PID Control in Mobile Robots with Embedded DSP," in *2008 International Conference on Intelligent Computation Technology and Automation (ICICTA)*, Hunan, 2008.

[38] Y. F. Y. Z. Y. H. K. Liu, "Accurate Speed Control for High-power Servomotor Based on Adaptive Backstepping Control Approach," in *International Conference on Mechatronics and Automation*, Harbin, 2007.

[39] N. B. S. O. Wahyunggoro, "Development of fuzzy-logic-based self tuning PI controller for servomotor," in *10th International Conference on Control, Automation, Robotics and Vision*, Hanoi, 2008.

[40] S. H. H. B. Hong Suh, "Proportional-Integral Plus Bang-Bang Control of DC Servo Motors with PWM Drives," *9th IFAC World Congress: A Bridge Between Control Science and Technology,* vol. 17, no. 2, pp. 2809-2813, 1984.

[41] A. Mellas, "Servo error amplifier gain compensation network," *IRE Transactions on Automatic Control,* vol. 7, no. 1, pp. 77-79, 1962.

[42] M. S. Sven Behnke, "Digital Position Control for Analog Servos," Humanoid Robots Group, Computer Science Institute, Freiburg, 2015.

[43] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology,* vol. 148, no. 3, pp. 574--591, 1959.

[44] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics,* vol. 5, no. 4, pp. 115--133, 1943.

[45] S. C. Kleene, "Representation of events in nerve nets and finite automata," RAND PROJECT AIR FORCE, SANTA MONICA CA, 1951.

[46] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review,* vol. 65, no. 6, p. 386, 1958.

[47] D. O. Hebb, The organization of behavior: a neuropsychological theory, J. Wiley; Chapman & Hall, 1949.

[48] R. S. Sutton, "Temporal Credit Assignment in Reinforcement Learning.," 1985.

[49] K. Koch, J. McLean, . R. Segev, . M. A. Freed, M. J. Berry, V. Balasubramanian and . P. Sterling, "How much the eye tells the brain.," *Current Biology,* vol. 16, no. 14, pp. 1428--1434, 2006.

[50] . K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural networks,* vol. 1, no. 2, pp. 119--130, 1988.

[51] P.-E. Danielsson and . O. Seger, "Generalized and separable Sobel operators," in *Machine vision for three-dimensional scenes*, Elsevier, 1990, pp. 347--379.

[52] . J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence,* no. 6, pp. 679--698, 1986.

[53] J. . J. Gibson, "The perception of the visual world.," *Houghton Mifflin,* 1950.

[54] K. McGuire, G. . D. Croon, C. W. De, K. Tuyls and H. Kappen, "Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone.," *IEEE Robotics and Automation Letters,* vol. 2, no. 2, pp. 1070--1076, 2017.

[55] A. Giachetti, M. Campani and . V. Torre, "The use of optical flow for road navigation," *IEEE transactions on robotics and automation,* vol. 14, no. 1, pp. 34--48, 1998.

[56] H. Cox, "A demonstration of Taylor's theorem," *Camb. Dublin Math. J,* vol. 6, pp. 80--81, 1851.

[57] B. D. Lucas and . T. Kanade, "An iterative image registration technique with an application to stereo vision," *Vancouver, British Columbia,* 1981.

[58] M. H. M. B. M. M. Samuel, "A Review of some Pure-Pursuit based Path Tracking Techniques for Control of Autonomous Vehicles," *International Journal of Computer Applications,* vol. 135, no. 1, pp. 35-38, 2016.

[59] J. M. Snider, "Automatic Steering Methods for Autonomous

Automobile Path Tracking," Carnegie Mellon University, Pittsburgh, 2009.

[60] F. F. a. M. P. A. Carullo, "A Low-Cost Contactless Distance Meter for Automotive Applications," in *IEEE Instrumentation and Measurement Technology Conference*, Brussels, 1996.

[61] J. J. A. a. C. F. M. Parrilla, "igital signal processing techniques for high accuracy ultrasonic range measurements," *IEEE Transactions on Instrumentation and Measurement,* vol. 40, no. 4, pp. 759-763, 1991.

[62] 3Racing, "KIT-M4 3RACING SAKURA M 1/10 M Chassis 4WD 2018," 2018. [Online]. Available: http://www.3racing.hk/manuals/KIT-M4_Manual.pdf. [Accessed 9 May 2020].

[63] Roboteq, "Roboteq User Manual," 8 July 2019. [Online]. Available: https://www.roboteq.com/docman-list/motor-controllers-documents-and-files/documentation/user-manual/272-roboteq-controllers-user-manual-v17/file. [Accessed 11 May 2020].

[64] Y. LeCun, L. Bottou, Y. Bengio and . P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278--2324, 1998.

[65] C.-L. Cham and Z. B. Samad, "Brushless DC Motor Electromagnetic Torque Estimation with Single-Phase Current Sensing," *Journal of Electrical Engineering & Technology,* vol. 9, pp. 742-748, 2014.

[66] J. Chu and M. Benaissa, "Low Area Memory-Free FPGA Implementation of The AES Algorithm," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, Oslo, 2012.

[67] . A. H. Ismail, H. R. Ramli and M. H. Marhaban, "Vision-based system for line following mobile robot," *10.1109/ISIEA.2009.5356366,* vol. 2, pp. 642 - 645, 2009.

[68] M. T. P. V. J.K. Hedrick, "Control Issues in Highway Systems," *IEEE control systems,* vol. 14, no. 6, pp. 21-32, 1995.

[69] DARPA, "Grand Challenge Overview," 03 March 2008. [Online]. Available: http://www.grandchallenge.org/grandchallenge/overview.html. [Accessed 16 05 2020].

[70] R. On, "Race On – USC Self-Driving Car Competition," Race On, 23 February 2020 . [Online]. Available: raceon.io. [Accessed 16 May 2020].

[71] K. H. R. G. S. P. F. Ciarnoscki, "Improvement of a mobile autonomous robot to participate in sumo competitions," in *2015 IEEE 24th International Symposium on Industrial Electronics (ISIE)*, Buzios, 2015.

[72] A. K. R. C. R. C. S. A. Singh, "Study of 4 Wheel Steering Systems to Reduce Turning," in *International Conference of Advance Research and Innovation (ICARI-2014)*, Delhi, 2014.

[73] M. S. S. O. H. Tourajizadeh, "Modeling and Optimal Contol of 4 Wheel Steering Vehicle Using LQR and its Comparison with 2 Wheel Steering Vehicle," in *International Conference on Robotics and Mechatronics (IcRoM 2018)*, Tehran, 2018.

APPENDICES

TABLE III
CAR SPECIFICATIONS (ANDREW)

Parameter	Description	Value	Unit
m	Mass [1]	1.76	kg
T_p	Pinion gear tooth count	28	-
T_s	Spur gear tooth count	80	-
R	Final drive ratio	5.7	-
V_{batt}	Nominal battery voltage	10.8	V
$I_{discharge}$	Maximum discharge current	30	A
I_{charge}	Maximum charge current	6	A
C	Battery capacity	2500	mAh
k_v	Motor speed constant	1855	RPM/kV
I_{rated}	Rated motor current	22	A
V_{max}	Maximum motor voltage	8.4	V
v_0	Car no load speed [2]	7.89	ms^{-1}
v_{rated}	Rated speed [3]	7.10	ms^{-1}
τ	Rated wheel torque [4]	0.644	Nm
μ_s	Static coefficient of tyre friction	0.9	-
a	Maximum acceleration [5]	8.82	ms^{-2}

[1] Mass includes 20% contingency. [2] Speed assuming no losses. [3] Maximum speed at which rated torque can be achieved. [4] Torque available at wheels with motor at rated current. [5] Acceleration is limited by traction in this case.

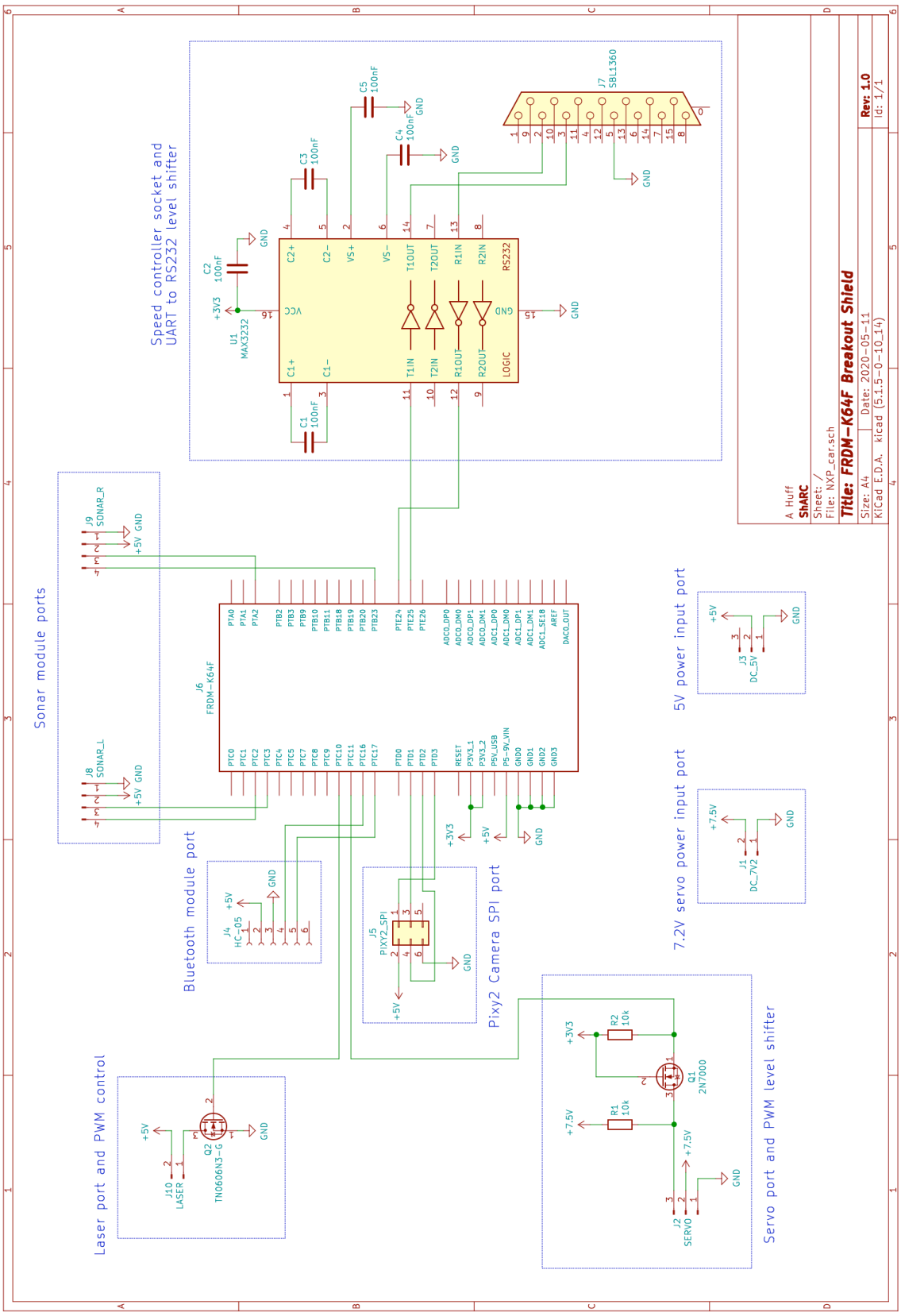

Fig. 18. (Andrew) Schematic of FRDM-K64F breakout shield.