**Project context**

The LUTZ finder pods are self-driving vehicles owned by the Transport Systems Catapult. These pods are drive-by-wire vehicles that rely on the input of around 19 sensors. CavLab is a facility which provides a suitable environment for the testing of these electric vehicles. However, during testing there must always be an engineer present within the vehicle to take over, via manual control, should it be necessary. The vehicles are quite large and expensive therefore testing these vehicles, on a regular basis, can prove quite costly. Consequently, Catapult have approached us to develop a μCavLab which will require the development of a much smaller drive-by-wire vehicle, at about 1:6 scale, which is compatible to the interface used in the original pod. The μPod will therefore be used as a representative initial testing model for testing the code and algorithms before testing is conducted on the full-size pod. The μPod will also be used as a demonstrator in STEM activities.

**Aims & Objectives:**

* Develop and design a miniature drive-by-wire vehicle design.
* Aim to design the vehicle such that it uses an Ackermann steering mechanism.
* Create a μPod such that it uses a drivetrain with a control unit which can be programmed in the same way as the full-size model.
* Aim to create the interface such that it will allow for other, more expensive, sensors to be attached to the vehicle.
* Use the μPod to test the code and algorithms originally developed for the full-size model.
* Aim to complete design architecture before the Christmas holiday such that there is enough time to build the vehicle.

**Project Plan:**

**2. Electrical & Software considerations**

On the basis of the mechanical design, the main task of electrical and software part is to focus on the establishment of a control system and execute it. In order to achieve the control system, the micro control unit and the sensor modules are two main component groups that should be integrated.

The micro control unit consists of a Raspberry Pi which plays as the main upper machine and an Arduino board which takes the role of the low position machine. The Raspberry Pi runs the Robot Operating System (ROS) - a [robotics middleware](https://en.wikipedia.org/wiki/Robotics_middleware) - to write robot software. This is capable of writing software with a wide range of tools, libraries and conventions which is easily integrated due to the modular design of the system. The ROS runs with its master-nodes model, processing publishing and subscribing topics between master - nodes or nodes - nodes. As the low position machine, Arduino is primarily charged with connecting nodes such as the gear motors, the GPS, the ultrasonic sensors etc. into the central control unit. To be more specific, the micro control unit works as the master to deal with the signals from various sensors and modules sent by wire.

The plan to complete this part will start with building a ROS environment on the Raspberry Pi, followed by building effective connection between Raspberry Pi and Arduino. After constructing the framework of the control computer and installing the components, connecting the motor(s) into the Arduino board is the next step. Once the connection has been made, we must program the ROS to control the vehicle to move as desired. Considering the obstacles on the road, we need to collect the effective traffic condition information from the sensors we have installed. Based on the information, cars should be controlled to make the right reactance such as turning to another direction when facing some obstacles. This avoidance function could be achieved by using ultrasonic sensors and LIDAR sensors. When testing the car, the remote control will be achieved by the Secure Shell (SSH) running on the user’s computer, remote logging in the Raspberry Pi to write program.