# KPI Review and Conclusions:

Note that the initial KPI from the project overview is bolded, and then conclusions drawn from these KPI are attached in non-bold.

* **Detection of proximity to ultrasonic sensor. Success for this criteria is described by having each ultrasonic sensor correctly display the distance it is reading on its user interface.** This criteria was successfully met, with the ultrasonic distance measured and successfully displaying on the console terminal from the driving zephyr program.
* **Timing synchronisation of both ultrasonic sensors, specifically the ability for both ultrasonic sensors to detect changes in distance in relatively the same amount of time (required for gesture control). Success for this criteria is described by successfully measuring both sensors adapting to a significant change in distance within 0.5 seconds of each other.** This criteria was successfully met, after overcoming some difficulties. Specifically, the initial implementation saw the left ultrasonic sensor “starving out” the right ultrasonic sensor in terms of the amount of data received by the nucleo board. However, this was circumvented by reordering task priorities of both reception of ultrasonic sensors.
* **The ability to enter gesture control mode. Specifically, the press of a button to switch modes and the representation of the system being in "gesture mode" being reflected in the UI.** This criteria was successfully met, though representing the system being in “gesture mode” in the UI was not implemented – as this was seen as unnecessary cluttering of the UI and adds unnecessary communication between the devices. Instead, due to well-implemented button debouncing, any press of the user button on the nucleo board will toggle the mode – and the device will always start in normal mode, so the mode of operation can always easily be tracked.
* **Motional capabilities of turtlebot, driven generally by the ultrasonic. Specifically, this criteria just requires each wheel pair of the turtlebot to move if the ultrasonic detects a "non-infinite" distance, and stop moving if no distance is recorded.** This criteria has been successfully met, as the turtlebot’s moves if distances are read, and are ignored if the distance read is 0 (which indicates that nothing was detected by the ultrasonic).
* **Speed control, modifiable by ultrasonic distances. Success is described by clear speed changes with increasing or decreasing the proximity to the ultrasonic sensor. Specifically, a smooth range of acceleration is experienced when slowly minimising the ultrasonic distance, up until maximal acceleration when the ultrasonic sensors read ~5cm. Further, response to these speed modifications must occur in real-time, with <1s of lag time permitted.** This criteria has been successfully met, as the TurtleBot clearly increases and decreases in velocity in accordance with ultrasonic distance measured, and this reading is smooth and occurs with minimal lag time.
* **Direction, dependent on the distance reading of each ultrasonic sensor. For example, if the left ultrasonic reads distances of ~5cm and the right ultrasonic reads no distances, then there is a maximal left rotation, but if the right ultrasonic's distance reading is increasingly decreased, the rotation will be less left. Success is defined by experiencing smooth increase in one-directional rotation when decreasing the distance readings of one ultrasonic sensor, and then a smooth decrease in this same directional rotation as the distance readings of the other ultrasonic sensor decreased. Further, response to these directional modifications must occur in real-time, with <1s of lag time permitted.** This criteria had to be significantly adapted given misunderstandings of TurtleBot operation. Controlling the TurtleBot’s motion came in the form of modifying its linear and angular velocity, not individually controlling each wheel. As a result, instead of having each ultrasonic sensor correspond to a wheel of the TurtleBot, one sensor corresponds to its linear velocity, and one corresponds to its angular velocity. With this new understanding, the rotation of the robot is smoothly and intuitively controlled by one of the ultrasonic sensors, and the overall intention of this KPI is still upheld.
* **Accurate detection of gestures (such as moving hands in certain patterns between ultrasonic sensors at a fixed distance of ~1m) and representation of this in the UI. Success for this criteria does not involve turtlebot motion, but rather the accurately UI display of the detected gesture.** This criteria has been successfully met, with the detected gesture being displayed in the bottom left corner of the M5Core.
* **Upon detecting a gesture, the turtlebot successfully moves in a way described by this gesture. Success for this criteria is initially described by the turtlebot accurately responding to one gesture type (whilst simultaneously ignoring other non-gesture ultrasonic readings) and then will be described by the turtlebot correctly responding to all ~5 of our gesture types.** This criteria has been met, with the communicated gesture reaching the TurtleBot and causing it to behave in the way indicated. For instance, once the M5Core displays that “Gesture 1” has been detected, the TurtleBot receives this communication and snakes from right to left.
* **Staying within the grid. Success for this criteria is described by the turtlebot automatically detecting that it has moved outside of the grid, and then turning around so that it can stay within the grid. Specifically, this means that it ignores gesture or ultrasonic input until the turtlebot is facing the direction to re-enter the grid.** This criteria has been successfully implemented, as the TurtleBot detects that it has moved outside of the 2x2m grid and immediately turns around, bypassing all current queued actions for it to perform.
* **Displaying on an M5Core2. Position of the turtlebot displayed in a UI representation of the grid. Also, LiDAR-only position displayed as an (x, y) coordinate, alongside the Kalman-filter fused position incorporating both LiDAR and velocity. Success is defined by accurately representing the turtlebot’s position in the grid, alongside displaying accurate measurements for fused and unfused positions.** The TurtleBot’s position is correctly displayed on the M5Core grid, which constitutes success for this criteria. However, the intention to display both fused and unfused positions has not been met. This is because the fused position was conclusively observed to be more accurate than the unfused position, so there is no point in displaying both positions – as this would just clutter the UI with unnecessary information.
* **LiDAR actually works to track position. LiDAR pings in a 360 direction off a surrounding boundary, with this reading being processed into a point cloud, which we use to locate the turtlebot’s location. Success is described by the LiDAR being able to accurately locate the position of the turtlebot within the grid (within a 20x20cm box of error).** This was correctly implemented, however, modifications to the LiDAR parameters and the processing of LiDAR data had to be made in order to restrict the “box of error”. The current implementation succeeds in minimising the “box of error” to the 20x20 benchmark highlighted in the KPI however.