EVGP Autonomous Safety and Control System Proposal

1. Introduction

The EVGP Autonomous safety and control system will be a three part system consisting of a Race Control System (RCS), a Master Safety Remote (MSR), and a Team Safety Remote (TSR).

The RCS will be a GUI computer based program where track officials can send controls to the car such as Race Start, Race End, and Individual or Group E-Stop. The RCS will connect to karts using a TCP socket connection over a WiFi communications network.

The MSR will be hardware provided to disable the drive motor controller on each kart. This simple On/Off relay will be a last resort stop for track officials to stop all vehicles that are not responding to the RCS.

The TSR can be used by teams at any point to individually stop their own kart in a similar fashion to the MSR.

2. Hardware

A. Race Control System (RCS)

Race officials shall set up a dedicated 2.4Ghz WiFi local area network (WLAN) for the cars.The network will have adequate broadcast power to cover the entire track with reliable connection to all karts in a Race. Teams are encouraged, but not required, to install a high-gain WiFi antenna on their kart for maximum reliability.

B. Master Safety Remote (MSR)

Teams will be provided a remote relay used to disable the kart. A central Master Safety Remote (MSR) controlled by track officials will be used to E-Stop all karts in the case the RCS fails (most likely due to a kart not responding).

The safety system should still function in the event that the team's primary computer or the RCS fails.This will take the form of the Master Safety Remote (MSR). The MSR should not be networked or controlled by a computer, instead being a simple direct communication between a wireless remote and a relay/contractor to cut power to the motor controller. The kart needs to have a way to detect if the remote is disconnected and will E-stop the kart (following the same state as the $RED_RED state in Section 3 A).

There are many affordable commercial off-the-shelf solutions for a remote relay.

C. Team Safety Remote (TSR)

Teams are encouraged to create their own safety remotes to individually E-Stop their karts. The remotes need to have programmable frequencies so that each team can have their own operating frequency.

If this safety remote is implemented, during the race the kart must act according to the stricter of the two messages from the RCS and the TSR. The kart cannot be controlled in any way other than E-Stop functionality by the TSR during a race.

3. States

A state is broadcast as a command to karts. Karts are expected to follow the state descriptions but no explicit implementation for a state is required. State commands are sent from the RCS according to RCS Commands.

A normal Race follows the sequence: IN_GARAGE -> GRID_ACTIVE -> GREEN_GREEN -> RED_FLAG -> RED_RED -> IN_GARAGE.

These commands will be sent to all karts and individual karts as explained below.

The track officials are expected to ensure reasonable safety of karts as defined in Aborting a Race.

A. State Descriptions

- IN_GARAGE: The kart is not currently driving in the Race.
    - All karts will be in this state as they connect to the RCS.

    - Karts are under control of teams for testing or race preparations.
    - Karts will be in this state while teams move their karts onto the track and are booting autonomous systems.
    - Karts not in the current race will be in this state if they are not in the RED_RED state, while other karts are racing.
- GRID_ACTIVE: The race has not started but will start soon.
    - All humans must be off the track before this command is sent. The kart should be ready for the race to begin within 2 minutes.
    - When ready to race, karts must reply with *$GRID_ACTIVE;* message to the RCS.
    - RCS will not send the *$GREEN_GREEN;* command until all karts have replied with a *$GRID_ACTIVE;* message.
    - If the track officials decide a kart is causing a delay by not sending the *$GRID_ACTIVE;* message, they may Abort the Race and take appropriate actions.
- GREEN_GREEN: The Race has started or is ongoing
    - Upon receiving the *$GREEN_GREEN;* command, a kart should begin driving within 30 seconds otherwise track officials are expected to issue a *$RED_FLAG;* command or Abort the Race.
    - The kart should acknowledge the first *$GREEN_GREEN;* command.
- RED_FLAG: The kart should begin slowing down to a stop while continuing to steer to avoid obstacles.
    - This command is used to allow karts to safely come to a stop while avoiding obstacles. This should not be a sudden complete stop in case there are other karts nearby.
    - Karts are encouraged to safely pull to the inside of the track to leave the race line clear.
    - 10-15 seconds is the expected amount of time to come to a stop from racing speed.
    - This command will be sent to individual karts shortly after finishing the race and can also be sent if an Incident occurs.
    - The kart should remain stopped once it has stopped moving and send a *$RED_FLAG;* message to the RCS.
- RED_RED: The kart must immediately stop as quickly as possible.
    - The kart should brake to stop.
    - This command may be sent to ALL karts after an Incident
    - This command will be sent after ALL karts have finished the Race whether or not they already received a *$RED_FLAG;* command.
    - Karts should reply with a *$RED_RED;* message after coming to a complete stop (Nominal speed of 0.0). If all karts have not sent the *$RED_RED;* message within a reasonable time, track officials should use the MSR even if all karts appear to be stopped.
    - After humans are allowed on the track, they must stop any autonomous programs before disconnecting from RCS. See System shut-down.

Humans may only be on the track with karts during any of the following conditions:

1. IN_GARAGE is being broadcast by RCS to every connected kart and track officials have cleared teams for the track.
2. All karts connected have Acknowledged the STOP message and track officials have cleared teams for the track.
3. The track officials have used the Master Safety Remote and cleared teams for the track.

B. System Start-up

Before a Race begins, the RCS shall broadcast *$IN_GARAGE;* while karts are connecting. When all karts in the Race are on the track and all humans have left the track, the RCS will broadcast *$GRID_ACTIVE;* until all In Race karts have Acknowledged the message. The RCS will not permit the race to be started until all karts in the race have acknowledged the GRID_ACTIVE command.

C. System Shut-down

When a kart has finished the required number of laps, the track official will mark the kart as Finished in the RCS. The RCS will send the kart a *$RED_FLAG;* command as described in the RCS Software description. When all karts have finished the required laps, the RCS will transition the Race to Race Over. The RCS will then broadcast the *$RED_RED;* command until all participating karts have Acknowledged. When all karts have Acknowledged and track officials deem the track safe for humans, a *$IN_GARAGE;* command will be sent to all karts and teams may enter the track.

Once the track is safe to enter, teams may disconnect from the RCS and take control of their karts. For safety, all karts must default to an ESTOP or stopped state upon disconnecting from the RCS to prevent the kart from automatically starting up.

D. Aborting a Race

In the case that a race needs to be aborted, track officials will stop all vehicles in the manner most appropriate to protect primarily human life and secondarily the karts. If humans are in immediate danger, the MSR should be used. If only karts are in danger, it is up to track officials to send either the *$RED_FLAG;* or *$RED_RED;* commands through the RCS software, or use the MSR to kill all the karts.

E. Disconnections

If a kart has disconnected (as described in Connection status) during the race, the kart shall immediately enter the RED_RED state. If the RCS determines that a kart has disconnected, the RCS will notify the track officials. It is up to track officials to end the race in a safe manner following procedure for Aborting a Race.

F. Karts not in the Race

The RCS will be provided with a list of Karts participating in the race during start-up. Any Kart that connects to the RCS that is not in the race will be kept in the IN GARAGE state and will only receive the message *$IN_GARAGE;*

4. Race Control System (RCS) Software Description

The RCS is designed to allow the track officials to control karts both collectively and individually, providing an intuitive interface that promotes track safety. The Georgia Tech RoboJackets has offered to build the RCS software as follows. The RCS will be written in Python, utilizing the socket library for TCP communication.The RCS computer will be a single PC with a static IP address and specific port number to guarantee all competitors are communicating with the correct PC.

The karts will connect to the RCS through a central server via a TCP socket over a WiFi communications network. The RCS GUI will allow track officials to track the state of all karts, add karts to a Race for management, initiate emergency stops, and control the global race state. The GUI will display karts as kart numbers as well as provide the command functions as noted below. Individual commands will be next to team names, and a section will be shown for commands that work for All karts or In Race karts. Status messages will be shown next to the karts in case of disconnection from the RCS or other general failures.

The RCS will be open-sourced. The version that is being developed by Georgia Tech is found here: https://github.com/RoboJackets/evgp-rcs.

A. Communications to RCS

The kart must be able to connect to and send/receive messages to the RCS through the Transmission Control Protocol (TCP) over a WiFi network. The TCP socket connection will be used by the RCS to control cars as described below. All karts will be assigned a static IP address corresponding to their team number for use with the RCS.

Because of the 1-to-1 connection nature of TCP, messages that are directed at a specific kart will only be available to that specific kart. If there is a message that applies to more than one kart at a time, each individual TCP connection will receive the same copy of the message on its specific connection.

TCP is chosen because it is a well-known, reliable communication system and many accessible programming libraries implement it for teams to use. There are many openly available libraries for working with TCP in most languages and many tutorials are available for sending the required characters through the TCP socket.

B. RCS Commands

The RCS has the following commands (as buttons) for track officials following the description set forth in State descriptions. The RCS GUI buttons will react as follows:

- ALL KILL (All Karts)
    - Send *$RED_RED;* command to All karts
    - Karts should send a $*RED_RED*; message back to the RCS once they have stopped moving.
- RED_FLAG (ONLY Karts Currently In Race)
    - Sends *$RED_FLAG;* to In Race karts.
    - Karts should send a *$RED_FLAG;* message back to the RCS once they have stopped moving.
- RED_RED (Individual)
    - Send *$*RED_RED*;* to the Individual kart
    - Karts should send a $RED_RED; message back to the RCS once they have stopped moving.
- GRID_ACTIVE (Only Karts Currently In Race)
    - Send *$GRID_ACTIVE;* to In Race karts
    - All karts must send the *$GRID_ACTIVE;* message to continue to GREEN_GREEN
- GREEN_GREEN (Only Karts Currently In Race)

    - Send *$GREEN_GREEN;* to karts In Race
    - Requires GRID ACTIVE to be the last button pressed and every kart In Race to send the *$GRID_ACTIVE;* message to the RCS.
- FINISH (Individual)
    - Marks that a kart has finished on the RCS, wait set amount of time, then send the *$RED_FLAG;* command.
    - The set amount of time will depend on what place the kart finished to stagger karts and avoid a crash at the finish line.
    - Karts should send a *$RED_FLAG;* message back to the RCS once they have stopped moving.
- RACE OVER (Only Karts Currently In Race)
    - Sends *$RED_RED;* to In Race karts
    - Will notify track officials when every In Race kart has sent the *$RED_RED;* message and then broadcasts IN_GARAGE to In Race karts.

The expected race sequence used is: GRID_ACTIVE -> GREEN_GREEN-> FINISH -> RACE OVER
In case of Race Abort: GRID_ACTIVE -> GREEN_GREEN -> RED_RED
In case of Race Incident during Race: GRID_ACTIVE -> GREEN_GREEN-> RED FLAG -> (After karts are in safe position) RED_RED

C. Message Format

All messages between karts and the RCS will be sent over a TCP socket in plain ASCII characters. The RCS will send the race state at least 10 times per second.
Each message will begin with a start character (a dollar sign: $) followed by a character byte array of the State name and finished with an end character (a semi-colon: ;)

Example: $GRID_ACTIVE; commands the GRID_ACTIVE state.

When a kart is required to send an acknowledgement back to the RCS as defined in States, it should echo back the received command in plain ASCII characters. For example, to acknowledge a *$GRID_ACTIVE;* command, a *$GRID_ACTIVE;* message will be sent back to the RCS.

D. Connection Status

The RCS will continuously determine if a kart is connected and will follow the Disconnected protocol if the connection is lost.The RCS will deem a kart disconnected when the Python Socket library used raises an exception during a send operation signalling a lost connection.

The command messages are sent multiple times a second because a failed send is the most reliable way to determine that the kart has disconnected from the TCP socket.