Guardian Angel

Name: Kavya Chandrika Vempalli

ASURITE: kvempall

ASU ID: 1229837858

Our version of guardian angel is a personalized context-aware application aimed at improving the quality of life and safety of individuals with diabetes, enhancing rider safety and emergency response. The project's primary objective is to develop a comprehensive system that monitors and manages blood sugar levels in real-time and enhances rider safety through an autonomous vehicle advisory controller.

This project includes,

- ➔ Emergency Services Notification
- ➔ Enhanced Rider Safety Measures
- ➔ Health-Centric Food Recommendations
- ➔ Tailored Accommodation Suggestions
- ➔ Reliable Advisory Controls.

The project is divided into three parts.

Section 1: In this section, user information is taken through a mobile application. User Information typically includes Heart Rate, Respiratory Rate, Blood Sugar levels, User Schedules, User's Travel Bookings and Medical Records. All this data is stored in the database.

Section 2: In this section, user's vehicle and travel information is taken. Vehicle information comprises of distance between two cars, Speed of the car. Travel information comprises of source and destination information of the travel and Road Condition.

Section 3: In this section, Advisory Controller is used. It takes the heart rate, respiratory rate and vehicle information from task 1 & task 2 and calculates whether the user has to switch or not. If the output is no switch, then it suggests a prompt "Relax", if the ouput is switch, it throws an alert through the mobile app. If the output is crash, emergency services are contacted. The Integration is done in project 5.

Alignment with Guardian Angel:

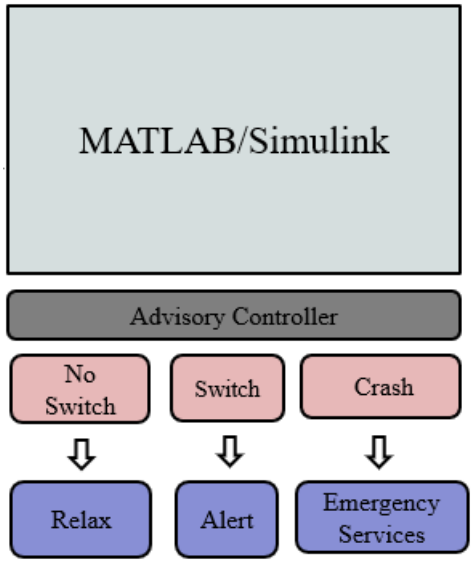
Created advisory controller using the user and vehicle data and determined if the user needs to be alert or if there is any emergency or if he can relax. Within project 4, I concentrated on the MATLAB segment of the implementation. To gather user data from a mobile app, a channel was established in thingsSpeak, encompassing six fields: HeartRate, RespRate, Speed, Distance, RoadCondition, and Collision_status. The initial five fields serve as inputs for the .m file, aiding in the computation of reaction time based on road conditions. LaneMaintainSystem.slx employs the initial Speed and Distance values to determine the speed limit and distance limit parameters.

During model simulation, distance and time series values are produced. These values are instrumental in calculating the time of collision. If all distance values generated exhibit

negativity, no immediate action is necessary, and the user can remain relaxed, denoted by a collision status of 1.

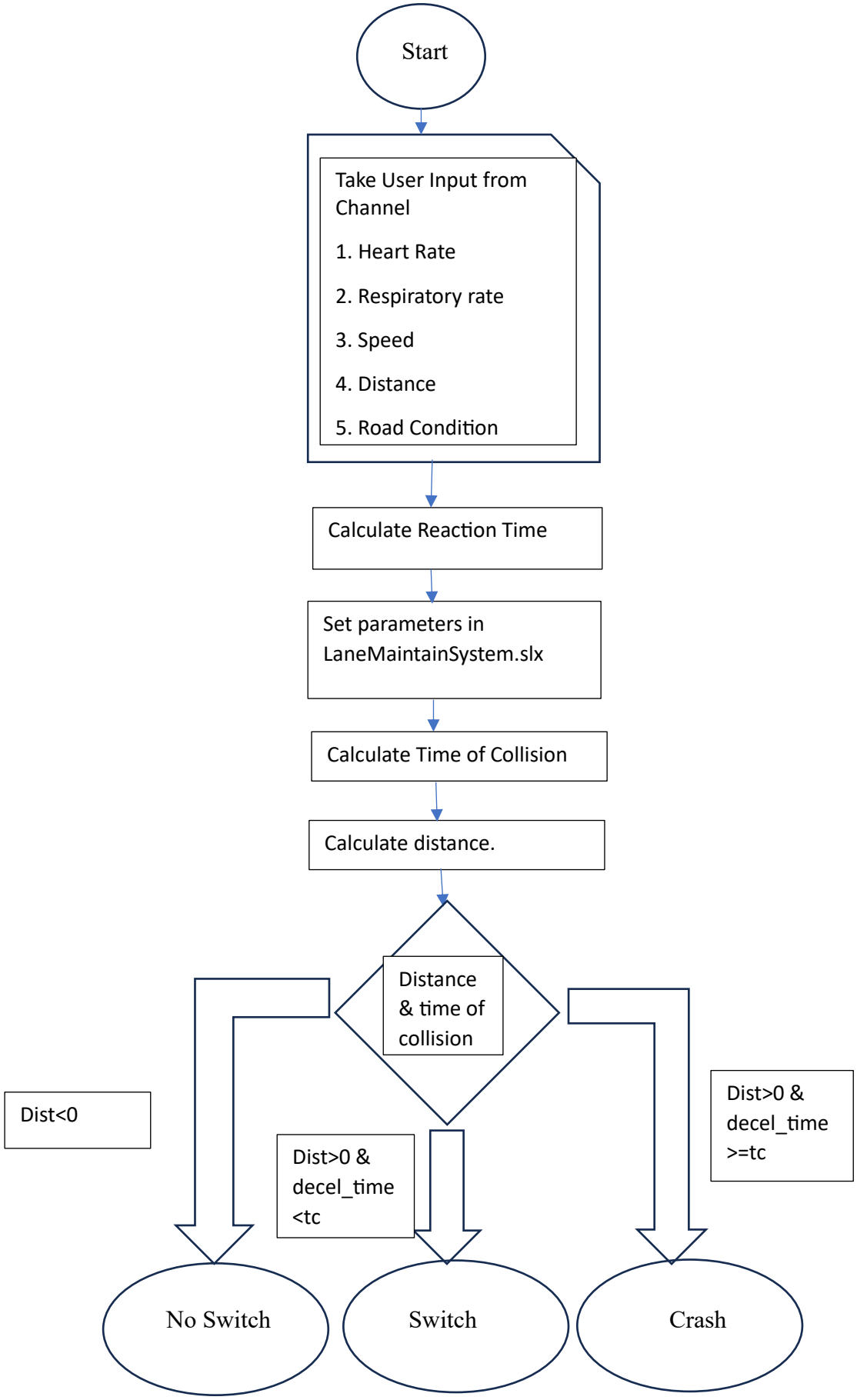
However, if some distances surpass zero, the HumanActionModel is executed to evaluate the feasibility of switching based on reaction time. If the duration required to decelerate the car is shorter than the time of collision, a switch is deemed possible (collision status 2). Conversely, if deceleration isn't completed before the impending collision time, a crash occurs, marked by a collision status of 3. The resultant output is channelled back for access by the mobile application.

Specifications



- For taking the user data from mobile app, a channel is created in thingsSpeak. The Channel consists of total six fields, HeartRate, RespRate, Speed, Distance, RoadCondition and Collision_status.
- The first five fields are taken as input by the .m file and depending on road condition, reaction time is calculated.
- Using the initial Speed and Distance values, the speed limit and distance limit parameters are LaneMaintainSystem.slx
- When the model gets simulated, it gives distance and time series values. Depending on the value of distance and time, time of collision is calculated.
- If all the distance values generated are negative, then there is no requirement to switch and the user can relax. The collision status is 1.
- If some of the distances are greater than 0, then the HumanActionModel is simulated to check if it is possible to switch or not, depending on reaction time.
- If the time taken to decelerate the car is less than the time of collision, then there is a possibility to switch. Here the collision status is 2. Else, the deceleration is not completed before the time of collision, then it results in crash. The collision status is 3 in this case. The output is given to the channel, to be accessed by the mobile application.

Design:


Start
Take User Input from Channel
1. Heart Rate
2. Respiratory rate
3. Speed
4. Distance
5. Road Condition
Calculate Reaction Time
Set parameters in LaneMaintainSystem.slx
Calculate Time of Collision
Calculate distance.
Distance & time of collision
Dist<0
Dist>0 & decel_time <tc
Dist>0 & decel_time >=tc
No Switch
Switch
Crash

Testing Strategies

1. Simulink terminal output checking
2. Changing gain values
3. Using scope to check crashes
4.Placing Human Action Model to check switch or no switch

Navigating Challenges

1. Tried Json output to give to mobile application
2. Tried MATLAB Web app and mobile app to connect with the mobile application
3. Tried APIs to link between mobile application and MATLAB