ME235 Project Report
Collision Avoidance for AV's using Real Time Location Systems

Sharnam Shah

May 2020

1 Abstract

In this project a simulation showing Collision Detection and Avoidance for 2 Autonomous cars using only Real Time Location measurements has been performed. When the cars come close to each other, the possibility of a collision is processed, detected fast and they are made to stop abruptly thereby averting collision.

2 Background and Motivation

Autonomous vehicles are a technology that would change the future of transportation. The current technological progress, investment and surge in R and D related to AV's has helped hasten their development process. Currently Level 5 Complete Autonomy is still to be achieved. Safety is one of the key issues that has yet to be established for AV's to become a reality. The developments in 5G would help decrease latency and speed up cloud access for AV's to communicate with each other.This could improve safety and help realize the AV mission. Real time location information transfer through the cloud could further help in improving safety along with other sensors. Real time location would be critical safety information as cars move at high speeds. Real time location systems (RTLS) have many other applications such as healthcare, Indoor location systems for shops, malls and factories. This makes a good case to explore RTLS.

3 Project Proposal Decisions and Challenges

The initial target of the project was to implement the most simplistic proof of concept case of vehicle RTLS and collision avoidance that models reality. In reality there would be a connected network of these cars transmitting various sorts of data and learning from each other. Given the constraints of the course, to show the proof of concept a scenario of having only 2 'effective cars' was thought of. One 'effective car' would be me walking/running around with a GPS. The second 'effective car' would be a moving location simulated from the cloud. To show the proof of concept,if these cars are going to collide, the cars would just be abruptly stopped rather than involving a path planning algorithm. The project was supposed to include the following - 1 MCU,1 GPS sensor, a cloud platform for storing location of both the cars in real time, a node and gateway to access the cloud. The location transmitting method to the cloud was to be chosen keeping in mind that the rate of cloud transmission should not take too long and should work well. Another technical challenge was the rate at which the GPS sensor obtains its position fix. The GPS should send signals sufficiently fast so that the AV can respond fast. While selecting the hardware components something that

could be interfaced relatively easily with PSoC and LabVIEW would be required. The capability of PSoC 5LP to connect to the gateway would also need to be checked.

4 Project Implementation and Challenges

4.1 Implementation Results and Summary

A simplistic simulation using 2 simulated car locations was performed and worked well. The GPS sensor was simulated keeping in mind actual fix rates, the LoRa gateway, the cloud were eliminated due to lack of time. The simulation included user inputs that enabled the user to set the speed and initial location of car 2. The results were as expected. When the cars were about to collide the red LED would turn ON, the loops would stop executing, the simulation would stop at that moment and the GUI would show the plot of the position of these cars till that moment.

4.2 Component Selection

A PARALLAX SIM33EAU GPS Module, 1 PSoC 5LP,1 STM B-L072Z-LRWAN1 LoRa MCU, LabVIEW and an AWS account was selected and bought.

While selecting the GPS the main considerations were a normal or fast GPS fix rate. An interface that would be simple to integrate with the PSoC and STM. For selecting the mechanism of transmitting data to the cloud, either Cellular networks or LoRa were the two main options as these are the only ones that work over a wide range without additional requirements. LoRa was selected as that is something I personally want to learn about (as it has many low energy IoT applications) and as integrating cellular networks would require another SIM card. An STM LoRa MCU was selected as STmicroelectronics has developed MCU's specifically for LoRa applications, the vast available online material for STM and its user interface. Please note that for the final demonstration the hardware could not be setup due to errors, the GPS was simulated and the cloud was removed.

Figure 1: The GPS and STM LoRa B-L072Z-LRWAN hardware

4.3 Implemented GUI

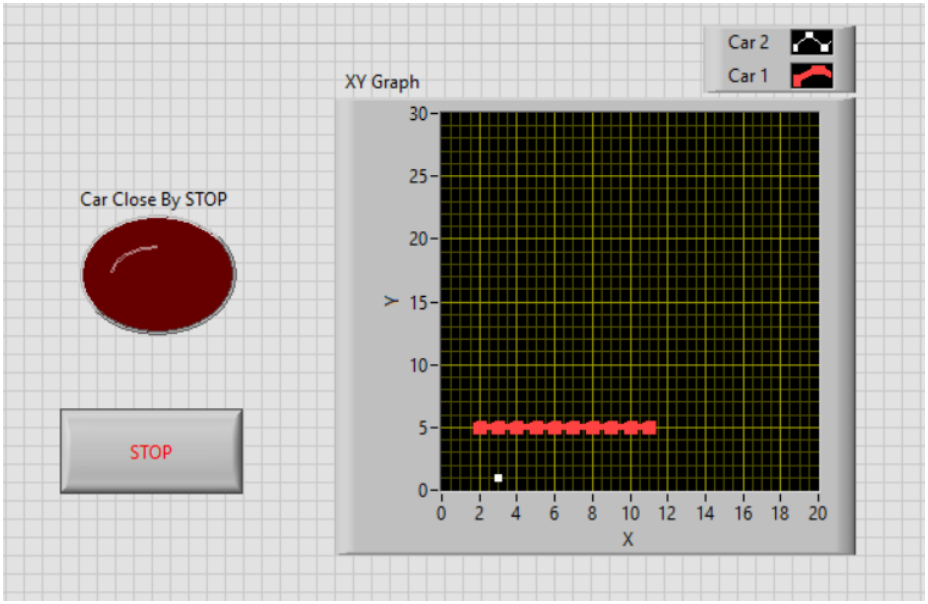

Figure 2: The GUI showing the cars in red, white and the emergency signal LED

The GUI seen in the final demonstration shows the trajectory of 2 cars, one in red and the other in white. The emergency Car close by LED lights up with Red colour when the distance between the 2 cars at any time instance becomes less then 2m. When the LED lights up, both the cars come to an abrupt stop, thereby avoiding collision. The user can enter and change parameters like car speed, position for car 2

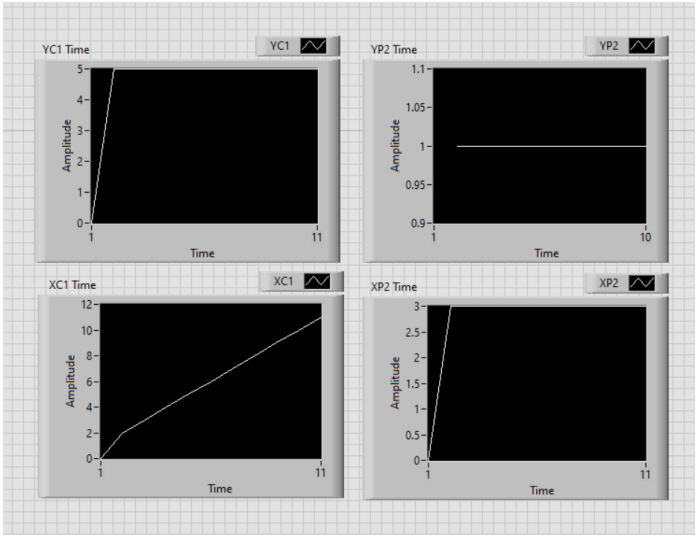

Figure 3: GUI Graphs showing the variation of the position coordinates in real time

4.4 Technical considerations, challenges and solutions

The project plan was to first set up the GPS, PSoC working and then depending on the time available, integrate the cloud. The integration of PSoC and GPS would be through the UART TX,RX and the parsing of GPS data would be implemented through the GetGPSLocation() function inbuilt in PSoC. One of the main challenges faced was to actually integrate the hardware and the cloud. The LorA, STM and cloud were not assembled due to lack of time. Reading GPS data from the to the PSoC did not work out. It seemed like the UART in the PSoC was not working, after my demonstration I learnt that the reason for this was different and would be solved by creating different PSoC UART pins.

Finally GPS data was simulated using a LabVIEW loop running every 1 second mimicking the fix duration. This LabVIEW loop would open, write and read from the PSoC every 1s. One of the 'effective car' locations was calculated from LabVIEW after the user inputs their speed preference in the GUI using this loop every second. For GPS location, to simply and make user friendly the X,Y coordinate were directly simulated from LabVIEW for car 2 and X,Y from PSoC for Car 1. In case Latitude and Longitude were to be simulated,the LabVIEW subVI would just include a few constants,multiplications and divisions to convert lat long to x,y. This can be done over small regions by directly using a formula. The other 'effective car' location was obtained from the PSoC.The PSoC was used to generate one location as in real scenario's the location would be directly read from the GPS by the PSoC. The PSoC location was updated every time LabVIEW wrote the other car's location to it, thereby maintaining synchronization. The PSoC location was set to increase with a fixed speed every time its read. Updating the PSoC location every 1s in synchronization with LabVIEW's 1s loop was done using the UART command interrupt that is activated once the entire UART message is read.

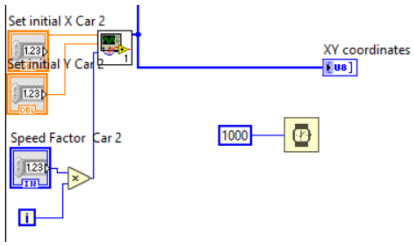

Figure 4: LabVIEW simulated GPS

```
50  CY_ISR(CommandReceived)
51  {
52      uint8 i;
53
54      if(CommandReady == 0)
55      {
56          Command_Packet.packet_size = ReceivedBuffer[0];
57          Command_Packet.command = ReceivedBuffer[1];
58
59          for(i=0;i<(Command_Packet.packet_size - 2);i++)
60          {
61              Command_Packet.buffer[i] =  ReceivedBuffer[i+2];
62          }
63
64          firstbyte = 0;
65          CommandReady = 1;
66          ByteCount = 0;
67      }
68      ByteCounter_ReadStatusRegister();
69      LEDDrive_Write(0);
70      //sum = sum+2;
71      stx = stx+1;
72  }
```

Figure 5: PSoC code showing Car 1 position (stx) update

An indicator loop was built within PSoC to detect the collision possibility. The cars would need to almost instantaneously stop when they come close to each other. In real scenarios, a car actuator would be directly controlled by the MCU. The indicator loop was chosen to be within the PSoC to enable the fastest feasible response to stop the car in real time. To measure collision possibility the squared of the euclidean distance between these two cars was checked every time the sensors update. When the distance is less than 2, an indication in PSoC (Car 1) is set that also passes to car 2 through LabVIEW. The overall data transfer cycle is as follows - after every 1s, LabVIEW writes car 2's location into PSoC, PSoC updates Car 1's location, implements the indicator distance and sends Car 1's updated X,Y, indicator flag value and Car 2's X,Y back to LabVIEW. LabVIEW then generates Car 2's new location and the cycle repeats. For transmitting all these locations from the PSoC the size of the transmit buffer was increased. A flag was chosen to indicate collision as it can directly be transmitted easily to LabVIEW and wired to a boolean Stop loop.

```
            }

            TransmitBuffer[Transmit_Packet.packet_size] = stx;
            TransmitBuffer[Transmit_Packet.packet_size+1] = sty;

            Transmit_Packet.packet_size = Command_Packet.packet_size+2;
            for(j=2;j<2+(TransmitBuffer[0]/2);j=j+2)
            {
            dist = ((TransmitBuffer[j] - stx)*(TransmitBuffer[j] - stx))+((TransmitBuffer[j+1] - sty)*(TransmitBuffer[j+1] - sty));
                if(dist<2)
                    {   //printf("%d",&diff);
                        flag = 100;
                        TransmitBuffer[1] = flag;
                    }
            }

            CommandReady = 0;
        break;
        default:
        break;
    }
    LabVIEW_UART_PutArray(TransmitBuffer,Transmit_Packet.packet_size);
}
```

Figure 6: PSoC code showing indiccator flag, euclidean distance and locations being written to UART

Another challenge faced was to use latitude, longitude and floating point data in PSoC. All data within PSoC is u8 format. The float would have to be represented after converting to u8 format. To avoid this extra conversion, data that was sent from LabVIEW was coerced to integer u8 format. The PSoC location generated was also an integer that can be represented in u8.

On the GUI side, to implement real time graphs, the waveform chart was used.This showed variation in real time. This chart continued for long without clearing prior data initially. To make the GUI well and show only current simulation data a 0 was wired to the history data property node for these charts and the simulation history time was set to 10 seconds. The XY graph showed the plot only after the simulation was stopped to avoid collision or after it was completed. A real time XY graph of the car positions was not implemented. To implement this, the position arrays would need to be stored after every LabVIEW loop iteration.

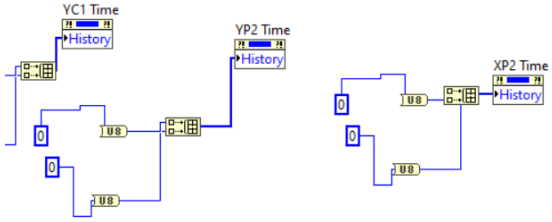

Figure 7: The wave form chart data being cleared - LabVIEW

5 Major Changes that may have Improved Project Outcome

Overall there are just two major changes that I think would have helped drastically improve the project outcome. First instead of thinking about th hardware for the cloud, checking whether STM supports interrupts and devoting time to the cloud, the focus should have been on how to use the PSoC, LabVIEW and GPS for the project.Setting that as the goal would have let me achieve better results. Spending more time on actually reading GPS data would have helped simulate something closer to reality. Second, some more time should have been spent on actually processing floating point values for position after converting them from lat,long to X,Y as this would again make the simulation more real. Otherwise the project seemed to be fine.