**PUBLIC TRANSPORT OPTIMIZATION**

**PHASE 3 SUBMISSION**

**TEAM MEMBERS:**

SANJAYKUMAR. S

NANDHAKUMAR. M

NISANTH. U

SHRIHARIHARAN. C

MANIVEL. S

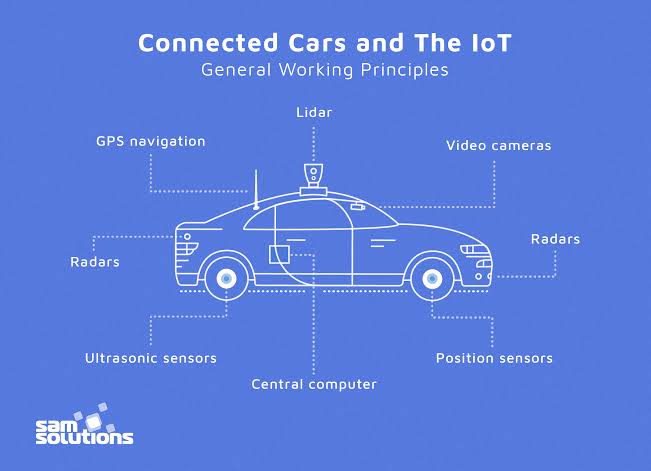
**MENTOR:**

PROF. PRABHU

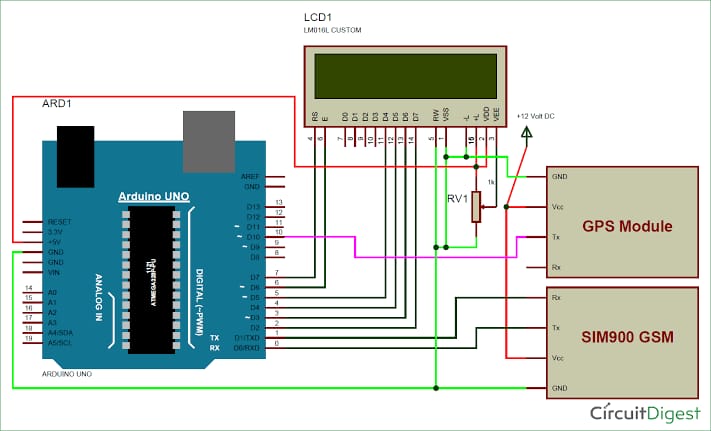
**PROJECT DESCRIPTION:**

GPS sensors in vehicles use satellite signals to determine the vehicle's precise location, speed, and direction. They're commonly used for navigation and tracking purposes. On the other hand, radar sensors in vehicles use radio waves to detect objects around the vehicle, helping with collision avoidance and adaptive cruise control. They're like eyes that help vehicles "see" their surroundings. Let me know if you'd like more details

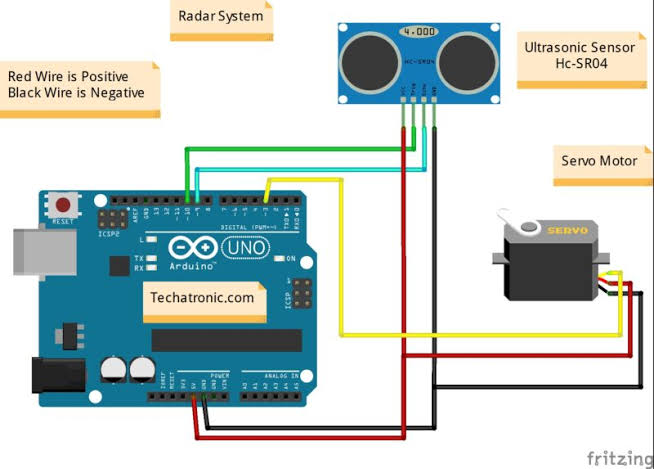
**PROJECT FUNCTION:**



**CIRCUIT DIAGRAM:**



**\*GPS SENSOR**



**\*RADOR SENSOR**

**CODING :**

% Initialize display for driving scenario example

helperAutoDrivingRadarSigProc('Initialize Display',egoCar,radarParams,...

rxArray,fc,vMax,rangeMax);

tgtProfiles = actorProfiles(scenario);

tgtProfiles = tgtProfiles(2:end);

tgtHeight = [tgtProfiles.Height];

% Run the simulation loop

sweepTime = waveform.SweepTime;

while advance(scenario)

% Get the current scenario time

time = scenario.SimulationTime;

% Get current target poses in ego vehicle's reference frame

tgtPoses = targetPoses(egoCar);

tgtPos = reshape([tgtPoses.Position],3,[]);

% Position point targets at half of each target's height

tgtPos(3,:) = tgtPos(3,:)+0.5\*tgtHeight;

tgtVel = reshape([tgtPoses.Velocity],3,[]);

% Assemble data cube at current scenario time

Xcube = zeros(Nft,Ne,Nsweep);

for m = 1:Nsweep

ntgt = size(tgtPos,2);

tgtStruct = struct('Position',mat2cell(tgtPos(:).',1,repmat(3,1,ntgt)),...

'Velocity',mat2cell(tgtVel(:).',1,repmat(3,1,ntgt)),...

'Signature',{rcsSignature,rcsSignature,rcsSignature});

rxsig = radar(tgtStruct,time+(m-1)\*sweepTime);

% Dechirp the received signal

rxsig = dechirp(rxsig,sig);

% Save sweep to data cube

Xcube(:,:,m) = rxsig;

% Move targets forward in time for next sweep

tgtPos = tgtPos+tgtVel\*sweepTime;

end

% Calculate the range-Doppler response

[Xrngdop,rnggrid,dopgrid] = rngdopresp(Xcube);

% Beamform received data

Xbf = permute(Xrngdop,[1 3 2]);

Xbf = reshape(Xbf,Nr\*Nd,Ne);

Xbf = beamformer(Xbf);

Xbf = reshape(Xbf,Nr,Nd);

% Detect targets

Xpow = abs(Xbf).^2;

[detidx,noisepwr] = cfar(Xpow,idxCFAR);

% Cluster detections

[~,clusterIDs] = clusterer(detidx.');

% Estimate azimuth, range, and radial speed measurements

[azest,azvar,snrdB] = ...

helperAutoDrivingRadarSigProc('Estimate Angle',doaest,...

conj(Xrngdop),Xbf,detidx,noisepwr,clusterIDs);

azvar = azvar+radarParams.RMSBias(1)^2;

[rngest,rngvar] = rngestimator(Xbf,rnggrid,detidx,noisepwr,clusterIDs);

rngvar = rngvar+radarParams.RMSBias(2)^2;

[rsest,rsvar] = dopestimator(Xbf,dopgrid,detidx,noisepwr,clusterIDs);

% Convert radial speed to range rate for use by the tracker

rrest = -rsest;

rrvar = rsvar;

rrvar = rrvar+radarParams.RMSBias(3)^2;

% Assemble object detections for use by tracker

numDets = numel(rngest);

dets = cell(numDets,1);

for iDet = 1:numDets

dets{iDet} = objectDetection(time,...

[azest(iDet) rngest(iDet) rrest(iDet)]',...

'MeasurementNoise',diag([azvar(iDet) rngvar(iDet) rrvar(iDet)]),...

'MeasurementParameters',{radarParams},...

'ObjectAttributes',{struct('SNR',snrdB(iDet))});

end

% Track detections

tracks = tracker(dets,time);

% Update displays

helperAutoDrivingRadarSigProc('Update Display',egoCar,dets,tracks,...

dopgrid,rnggrid,Xbf,beamscan,Xrngdop);

% Collect free space channel metrics

metricsFS = helperAutoDrivingRadarSigProc('Collect Metrics',...

radarParams,tgtPos,tgtVel,dets);

end

**CODING 2:**

import serial

import csv

# Open a connection to the GPS device

gps\_serial = serial.Serial('COM3', 9600, timeout=1)

# Create a CSV file for data storage

csv\_file = open('gps\_data.csv', 'w', newline='')

csv\_writer = csv.writer(csv\_file)

# Write header row

csv\_writer.writerow(['Timestamp', 'Latitude', 'Longitude', 'Altitude'])

try:

while True:

gps\_data = gps\_serial.readline().decode('utf-8')

if gps\_data.startswith('$GPGGA'):

data\_fields = gps\_data.split(',')

if len(data\_fields) >= 10:

timestamp = data\_fields[1]

latitude = data\_fields[2]

longitude = data\_fields[4]

altitude = data\_fields[9]

csv\_writer.writerow([timestamp, latitude, longitude, altitude])

except KeyboardInterrupt:

# Close the CSV file and serial connection on keyboard interrupt

csv\_file.close()

gps\_serial.close()

**Workflow Template:**

**Phase 1: Setting up the Work Environment**

* Install necessary Python packages
  + Install pandas
  + Install numpy
  + Install matplotlib
  + Install geopandas
* Set up a Python environment for coding #EnvironmentSetup

**Phase 2: Data Collection**

* Connecting to GPS device #DataCollection
  + Check GPS device connection
  + Configure GPS device settings
* Collect GPS signals
  + Start GPS data collection
  + Monitor GPS data collection process
  + Stop GPS data collection
* Save collected data into a file

**Phase 3: Data Processing**

* Load the collected data into Python #DataProcessing
* Convert GPS signals into readable format
  + Define the GPS signal structure
  + Parse the GPS signals
* Extract useful data from GPS signals
  + Extract longitude and latitude
  + Extract altitude
  + Extract timestamp

## Phase 4: Data Analysis and Visualization

* Analyze the extracted data #DataAnalysis
  + Calculate distance travelled
  + Calculate speed
* Visualize the data
  + Plot the GPS data on a map
  + Plot speed over time
* Save and export the results

### Note:

* Bulleted lists represent tasks that can be done in any order.
* Numbered lists represent tasks that should be done in a specific order.
* Checkboxes represent tasks that can be marked as complete or incomplete.
* Indentation is used to show hierarchy and relationship between tasks and sub-tasks.
* Hashtags are used for prioritizing, labeling, organizing, grouping, and tagging the tasks.

THANK YOU!