Drowsiness Detection

Advanced Mobile Computing Class

Team 5: Do Il Yoon, Hyun Seok Oh, Hyunik Kim, Phil Goo Kang

Table of Contents

1. Introduction
2. Schedule
3. Development Process

Introduction

- Driving is a common task done by countless people everyday.
- A big portion of accidents on road is caused by fatigued, drowsy drivers.

Characteristics to Drowsy Driving

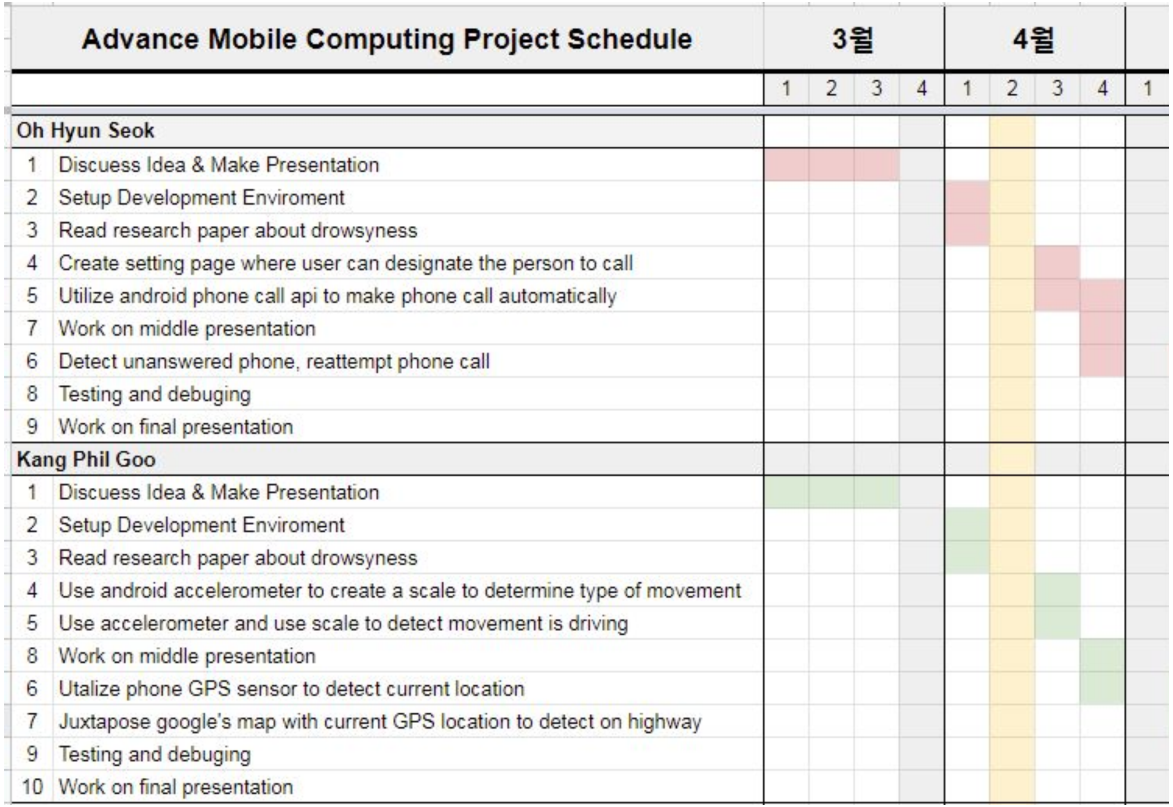
- Researchers found several characteristics linked to drowsy driving [1][2]:
  - Occur late at night (0:00 am–7:00 am) or during mid-afternoon (2:00 pm–4:00 pm)
  - Occur on high-speed roadways
  - Driver is often alone
  - Change in eye blink duration

Possibility of Sensing & Detecting Drowsiness from Driver

- By carefully observing drivers' seat:
  - Drive-aid smartphone applications are prevalent among drivers (ex. Google Maps, T-map)
  - Phones are attached in the place where they don't impede drivers' vision, faced towards drivers at the same time

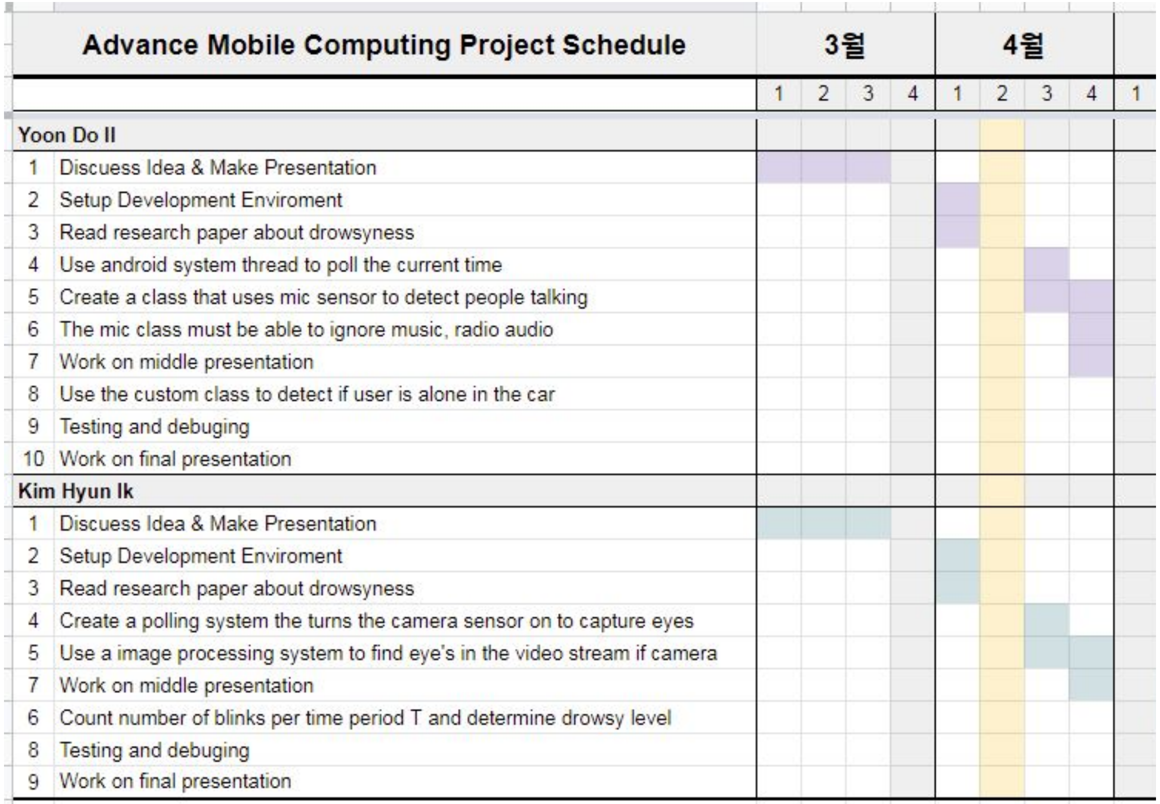
→ Possible to use front cameras, proximity sensors

Schedule and Role



Advance Mobile Computing Project Schedule		3월				4월				
		1	2	3	4	1	2	3	4	1
Oh Hyun Seok										
1	Discuess Idea & Make Presentation	🟥	🟥	🟥						
2	Setup Development Enviroment					🟥				
3	Read research paper about drowsyness					🟥				
4	Create setting page where user can designate the person to call							🟥		
5	Utilize android phone call api to make phone call automatically							🟥	🟥	
7	Work on middle presentation								🟥	
6	Detect unanswered phone, reattempt phone call								🟥	
8	Testing and debuging									
9	Work on final presentation									
Kang Phil Goo										
1	Discuess Idea & Make Presentation	🟩	🟩	🟩						
2	Setup Development Enviroment					🟩				
3	Read research paper about drowsyness					🟩				
4	Use android accelerometer to create a scale to determine type of movement							🟩		
5	Use accelerometer and use scale to detect movement is driving							🟩		
8	Work on middle presentation								🟩	
6	Utalize phone GPS sensor to detect current location								🟩	
7	Juxtapose google's map with current GPS location to detect on highway									
9	Testing and debuging									
10	Work on final presentation									

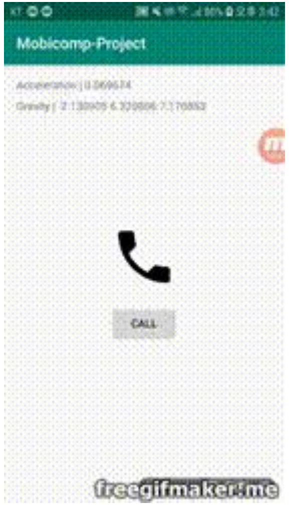
Schedule and Role



Advance Mobile Computing Project Schedule		3월				4월				
		1	2	3	4	1	2	3	4	1
Yoon Do Il										
1	Discuess Idea & Make Presentation	■	■	■						
2	Setup Development Enviroment					■				
3	Read research paper about drowsyness					■				
4	Use android system thread to poll the current time							■		
5	Create a class that uses mic sensor to detect people talking							■	■	
6	The mic class must be able to ignore music, radio audio								■	
7	Work on middle presentation								■	
8	Use the custom class to detect if user is alone in the car									
9	Testing and debuging									
10	Work on final presentation									
Kim Hyun Ik										
1	Discuess Idea & Make Presentation	■	■	■						
2	Setup Development Enviroment					■				
3	Read research paper about drowsyness					■				
4	Create a polling system the turns the camera sensor on to capture eyes							■		
5	Use a image processing system to find eye's in the video stream if camera							■	■	
7	Work on middle presentation								■	
6	Count number of blinks per time period T and determine drowsy level									
8	Testing and debuging									
9	Work on final presentation									

After-detection & UI

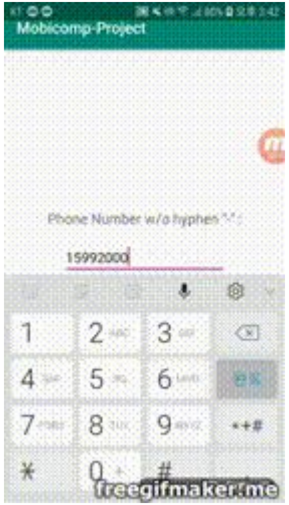
- Designated Peer calling
  - Call the peer when driver is classified as drowsy
    - Drowsiness Detector is not implemented, so currently press button to call
  - Can save peer phone number in setting
  - TODO : Android api cannot detect whether outgoing call is rejected or not responded, so should find ways to detour this
- UI design
  - Two tabs(Fragments), one for camera preview and other for settings



Mobicomp-Project

Acceleration | 0.069674

Gravity | 2.130905 6.320606 7.170852

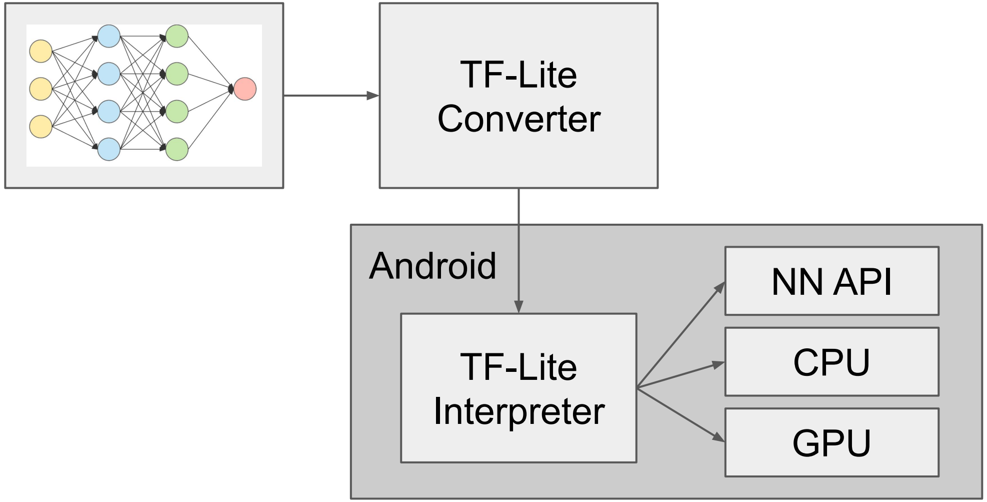


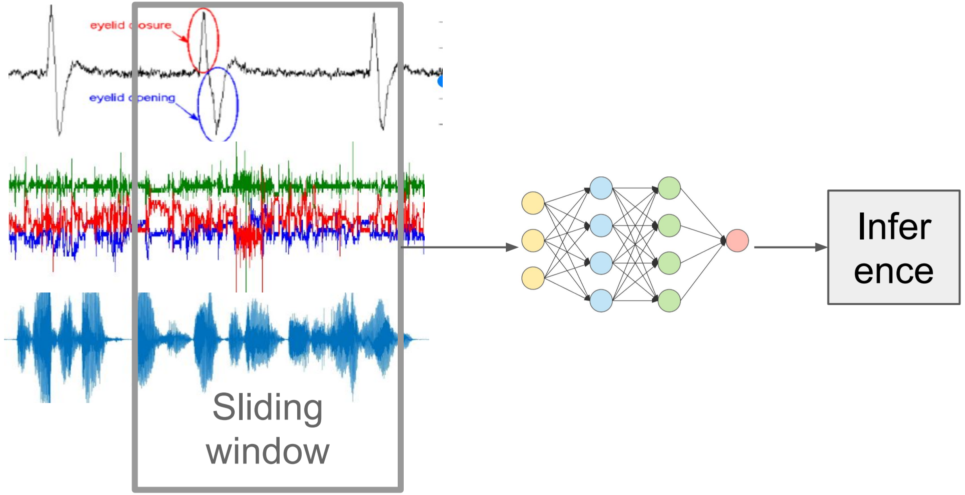
Mobicomp-Project

Phone Number w/o hyphen "-" :

15992000

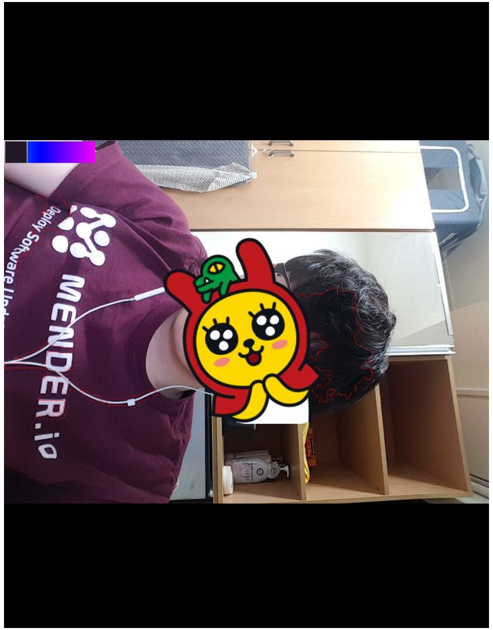
- Tensorflow Lite on detector module
    - Separate background thread for drowsiness detection
    - CNN module on sliding window input
        - Just for testing the system, not in work, model will change in the future
    - RNN or LSTM is almost impossible in TF Lite currently
    - Easiest and lightweight deep-learning framework for mobile inference.
- Attached other modules (Accelerometer, Location) to a whole system
    - Aim to attach audio and eye blinking data to the system





Eye Detection

- Used OpenCV Library to extract features from the front camera
- LBP classifier is used to detect face
- Haar classifier is used to detect eyes from the face
- Several Challenges:
  - Current detection causes too much overhead
  - Orientation Issue (Default: Landscape mode)
  - People wearing glasses



Integrating Accelerometer [1/2]

- Utilize smartphone native android accelerometer API
- Created background service (thread) to continually retrieve accelerometer sensor data
- Different types of sensor options:
    - TYPE_ACCELEROMETER: add options like offset and angle
    - TYPE_ACCELEROMETER_UNCALIBRATED: actual raw hardware value
    - TYPE_LINEAR_ACCELERATION: applies gravity and rotation calculations then returns value

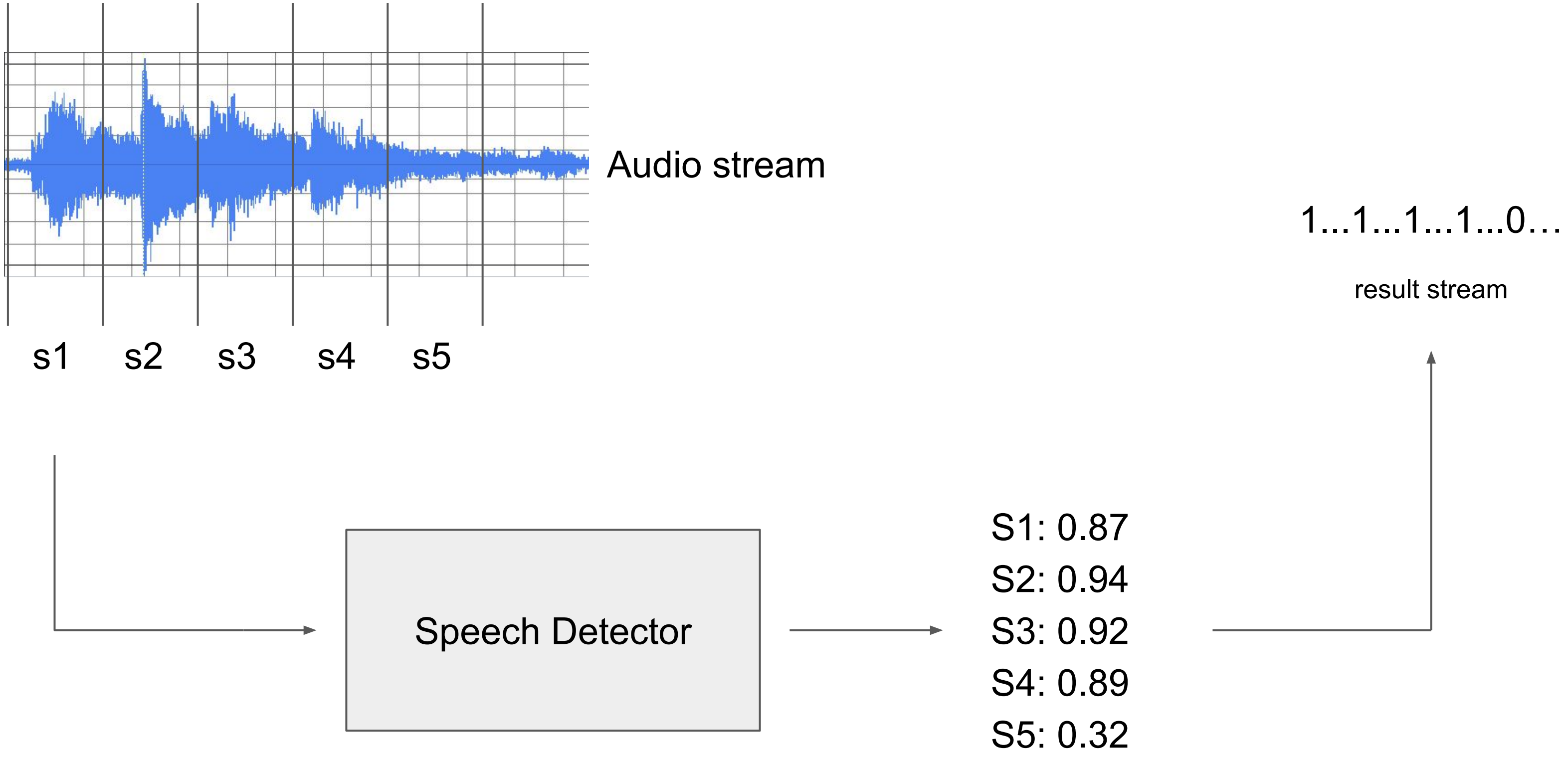
Integrating Accelerometer [2/2]

- Energy Efficiency:
  - Constantly using sensor drains battery very quickly
  - Use a polling method to reduce power drain (ex 10s)
  - TODO: Need additional research for best performance level
- Sensor options:
  - Android OS collects hardware sensor data via different types of algorithms
  - As mentioned before current test show TYPE_LINEAR_ACCELERATION as best option
  - TODO: Need to check for other types of data cleaning (preprocessing)

Integrating Location Tracker

- Two methods of user location tracking is available:
  - GPS: Global Positioning System
  - LBS: Networking based locating service
- Because user is inside of a car, we will be using both methods to improve accuracy
- Location tracking API only allows polling method to retrieve data
- We are testing using 1s as polling interval but through test we will try to configure best option
- TODO: Need to integrate both accelerometer and location tracking sensor data to help determine drowsing

Integrating Speaker Detection



Audio stream

s1 s2 s3 s4 s5

Speech Detector

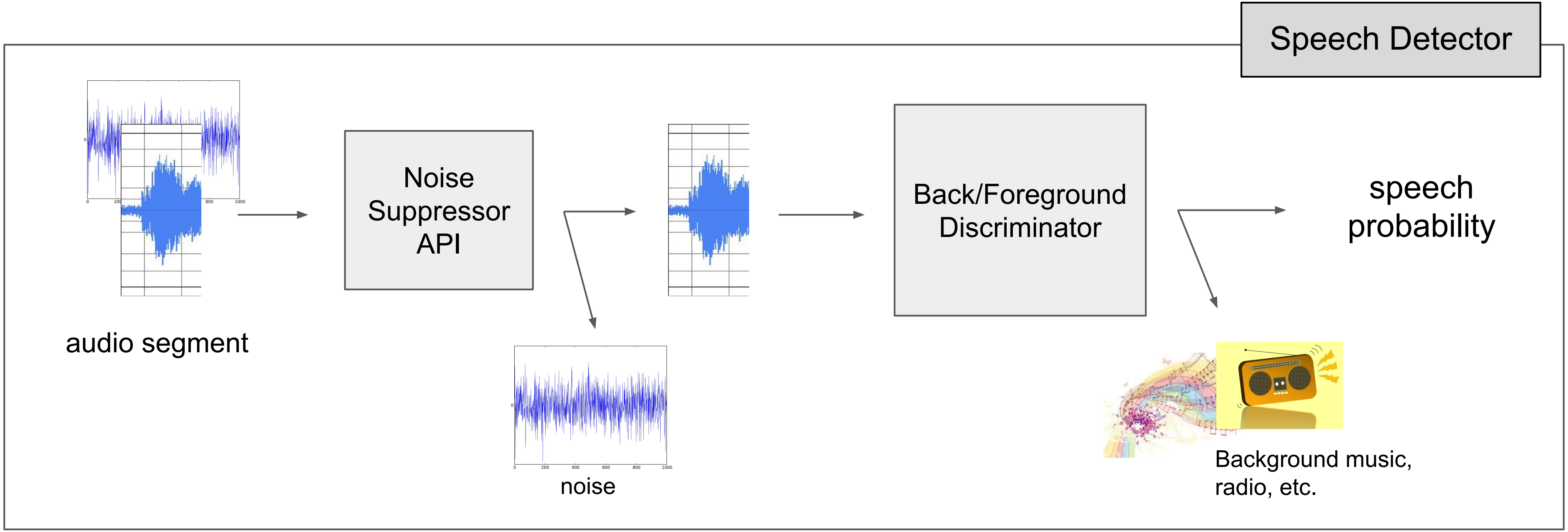
S1: 0.87
S2: 0.94
S3: 0.92
S4: 0.89
S5: 0.32

1...1...1...1...0…

result stream

Integrating Speaker Detection

- Use the MediaRecorder API to get audio data stream
- Pass the audio segments to the Speech Detector module


Speech Detector
audio segment
Noise Suppressor API
noise
Back/Foreground Discriminator
speech probability
Background music, radio, etc.