EE595 - Final Project

Team11

20213705 Yeongbin Hwang

Index

- PART1 (Trajectory Tracking)

- PART2 (Mobile Bodyguard)
  - motion recognition
  - voice recognition

Part 1
Trajectory Tracking

PART1

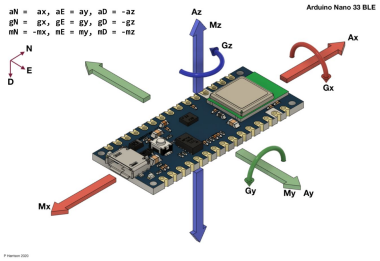
Algorithm for Trajectory Tracking

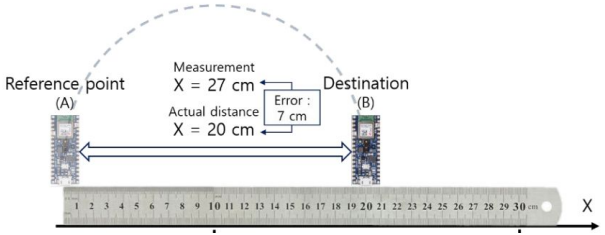
1. **Calculate the angle(roll, pitch, yaw) of Arduino**
   -using **Kalman Filter** and **Complementary Filter**

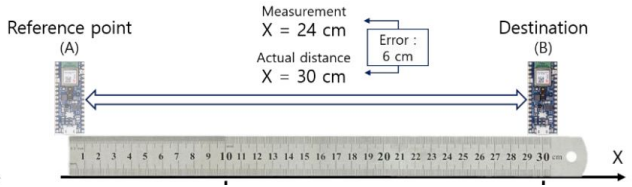
2. **Remove roll, pitch, yaw value to get accelerations with the absolute axis**

3. **Double integral to compute location**

4. **Considered the y-axis distance as the final trajectory**







PART1

Kalman Filter and Complementary Filter

Kalman Filter

- Use to fuse measured values from various noisy sensors
- Applied when the measured value of an object includes a probabilistic error

```
IMU.readGyroscope(gyroX, gyroY, gyroZ);
IMU.readAcceleration(accX, accY, accZ);

double dt = (double)(micros() - timer) / 1000000; // Calculate delta time


accXangle = (atan2(accY,accZ)+PI)*RAD_TO_DEG;
accYangle = (atan2(accX,accZ)+PI)*RAD_TO_DEG;
double gyroXrate = gyroX / 131.0; // Convert to deg/s
double gyroYrate = gyroY / 131.0; // Convert to deg/s

gyroXangle += gyroXrate*dt; // Calculate gyro angle without any filter
gyroYangle += gyroYrate*dt;


kalAngleX = kalmanX.getAngle(accXangle, gyroXrate, dt); // Calculate the angle using a Kalman filter
kalAngleY = kalmanY.getAngle(accYangle, gyroYrate, dt);
rotate_z = rotate_z + gyroZ * dt * RAD_TO_DEG;
```

Complementary filter

- gyroscope : drift occurs in the low frequency
- accelerometer : noise is generated in the high frequency
- Filter that complements each other.

```
IMU.readAcceleration(ax, ay, az);
angleAcX = atan(ay / sqrt(pow(ax, 2) + pow(az, 2)))/ (2*PI/360);
angleAcY = atan(-ax / sqrt(pow(ay, 2) + pow(az, 2)))/ (2*PI/360);

IMU.readGyroscope(gx, gy, gz);
gx = -gx; // X axis reverse
rotate_x = rotate_x + (gx - averGyX )* microTime_delta / 1000000;
rotate_y = rotate_y + (gy - averGyY )* microTime_delta / 1000000;
rotate_z = rotate_z + (gz - averGyZ )* microTime_delta / 1000000;

double angleTmpX = filtered_angle_x + rotate_x *  microTime_delta / 1000000;
double angleTmpY = filtered_angle_y + rotate_y *  microTime_delta / 1000000;
double angleTmpZ = filtered_angle_z + rotate_z *  microTime_delta / 1000000;

const float ALPHA = 0.96;
filtered_angle_x = ALPHA * angleTmpX + (1.0 - ALPHA) * angleAcX;
filtered_angle_y = ALPHA * angleTmpY + (1.0 - ALPHA) * angleAcY;
filtered_angle_z = rotate_z;
```

Part 2
Mobile Bodyguard

Function

Emergency situation

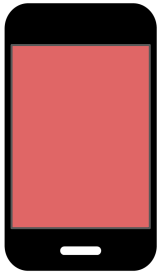
detect emergency situation

Function

Emergency situation

detect emergency situation

send signal (BLE)



Define Emergency Situation

How can we define an emergency situation?

Situation 1 :
Only Motion

Situation 2 :
Voice

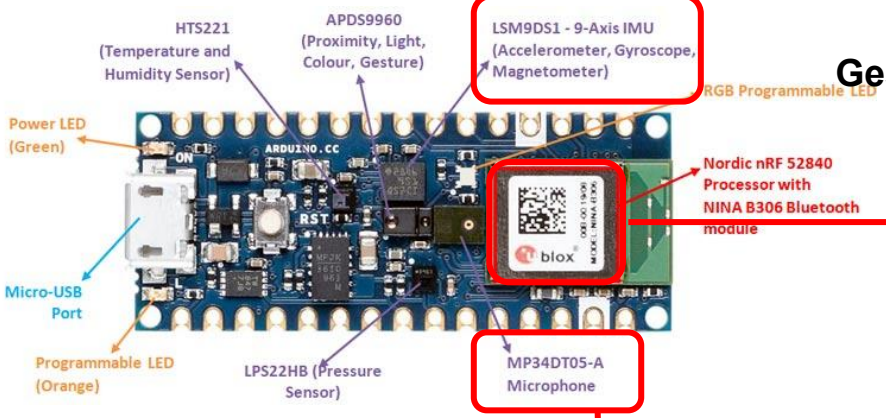
Define Emergency Situation

How can we define an emergency situation?

Situation 1:
Motion Recognition

Situation 2:
Voice Recognition

Implementation Overview



Step 1: Recognition

IMU

Gesture recognition

Microphone

Speech recognition

Step 2: Communication

Bluetooth module

BLE communication

Situation 1

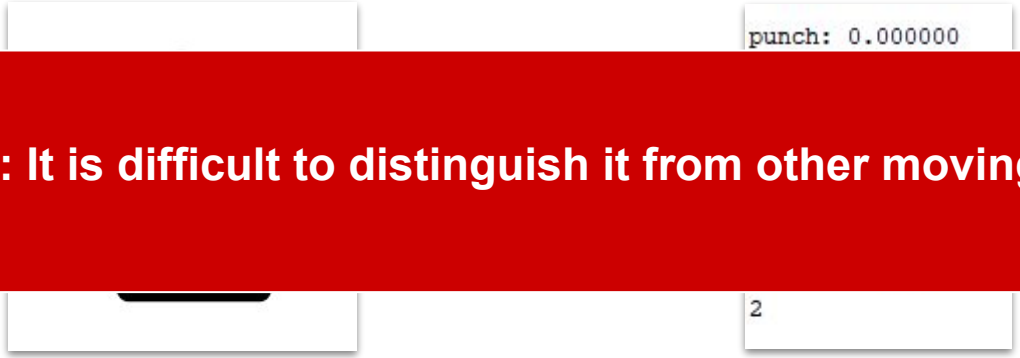
- Motion Recognition
  - Define the gesture to trigger
  - Train the model with 3 gestures (using colab).
    - features : 3 gyro + 3 accelerometer
    - gestures : **shake**, rotate, hands-up

aX	aY	aZ	gX	gY	gZ
-0.645	1.072	0.799	8.606	-43.335	-180.603
-0.378	0.896	1.103	59.631	-72.144	-270.325
-0.271	0.682	1.434	55.115	-73.181	-310.547
-0.317	0.355	1.257	28.748	-28.809	-322.998
-0.252	0.031	1.053	8.24	10.315	-308.167

Situation 1 :
Only Motion

Situation 1

- Motion Recognition
  - Define the gesture to trigger
  - Train the model with 3 gestures (using colab).
    - features : 3 gyro + 3 accelerometer
    - gestures : **shake**, rotate, hands-up

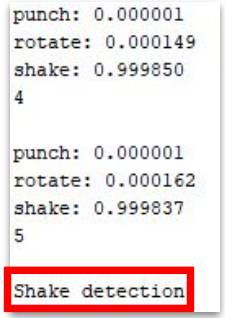


Challenge 1 : It is difficult to distinguish it from other moving motions.

Situation 1

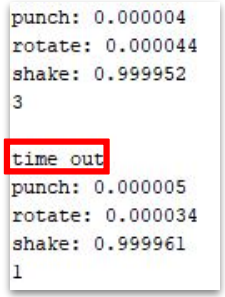
- Solution: avoid misrecognition with normal behavior.
  - Set the number of gesture to trigger

    => avoid false positive.



  - Set the reset time

    => detect only continuous 5 shaking.



Situation 2

- Voice Recognition
  - Define the voice to trigger
  - Train the model with 3 voices (using colab)
    - features : rms(root mean square) of PDM datas
    - voices : **help**, ok, no.

mic
627.696
685.159
782.087
712.867
749.908
706.754

Situation 2 :
Voice

Situation 2

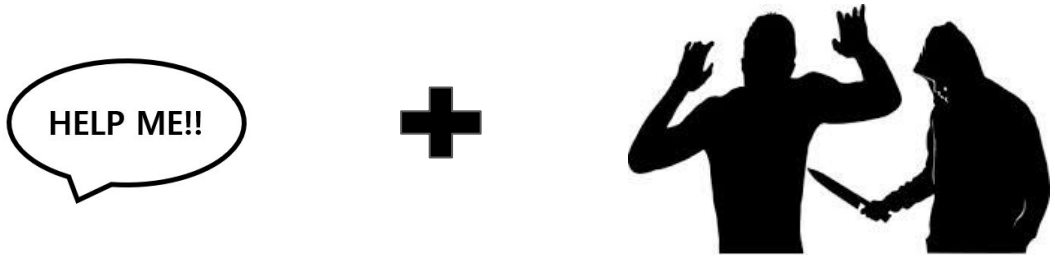
- Voice Recognition
  - Define the voice to trigger
  - Train the model with 3 voices (using colab)
    - features : rms(root mean square) of PDM datas
    - voices : **help**, ok, no.

Challenge 2 : misrecognition may occur even in model with very high accuracy

Situation 2

- Solution: using additional recognition
  - Voice + Motion
    - add **hands up** motion
    - after voice activation, recognize the motion.

  => more robust recognition.



Situation 2

- Solution: using additional recognition
  - Set the time window

    => detect only voice + motion.

```
// voice detection and hands-up
if (voice_detected && max_num == 0 && max_value > 0.9){
  emergency_detected = true;
  voice_detected = false;
}
```

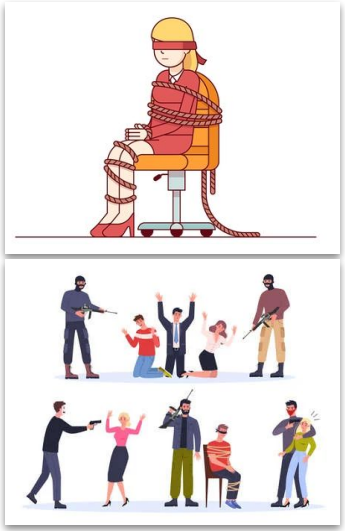
```
voice detection
Predicting the word:
Label 0 = 0.993017
Label 1 = 0.006983
Word detected: help

hands-up  0.999382
rotate: 0.000000
shake: 0.000618
0

emergency detection (voice + motion)
```

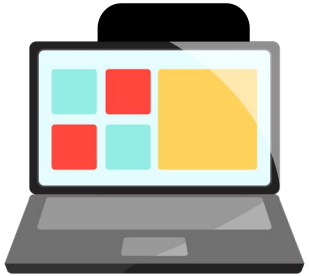
Emergency Alert

Emergency situation



automatically detect
emergency situation

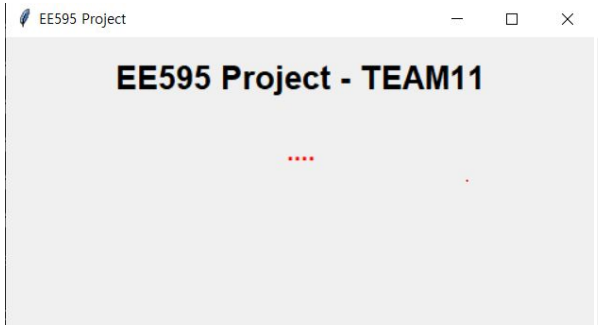
send signal (BLE)



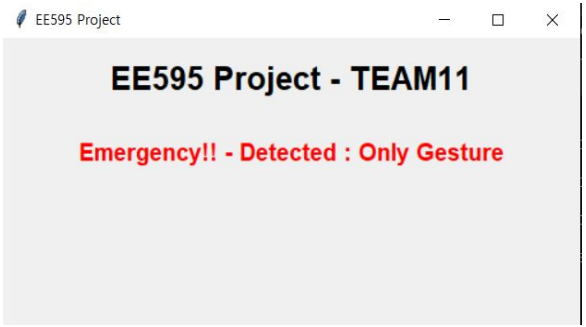
Emergency Alert using BLE

Emergency Alert

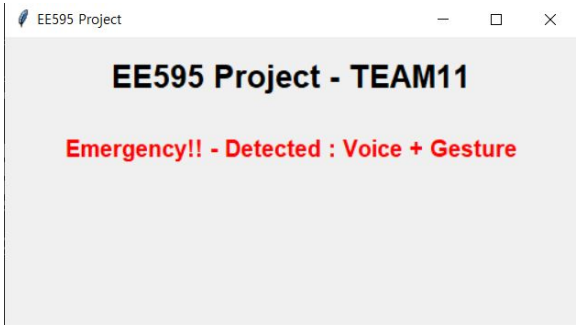
- Emergency Alert - BLE
  - Using BLE communication
  - Characteristics with “Notify”
  - Sending emergency situations and received data information



Normally



Detected Emergency Situation 1: Motion Recognition



Detected Emergency Situation 2: Voice + Motion Recognition

Performance

- Accuracy of Gesture recognition
  - repeat 50 times (shake, other gestures)
  - gesture : **shake**, hands up
  - shake
    - Precision = 89%
    - Recall = 96%
    - Accuracy = 92%

		answer	
		shake	**other gestures**
result	**shake**	48(TP)	6(FP)
	other gestures	2(FN)	44(TN)

Performance

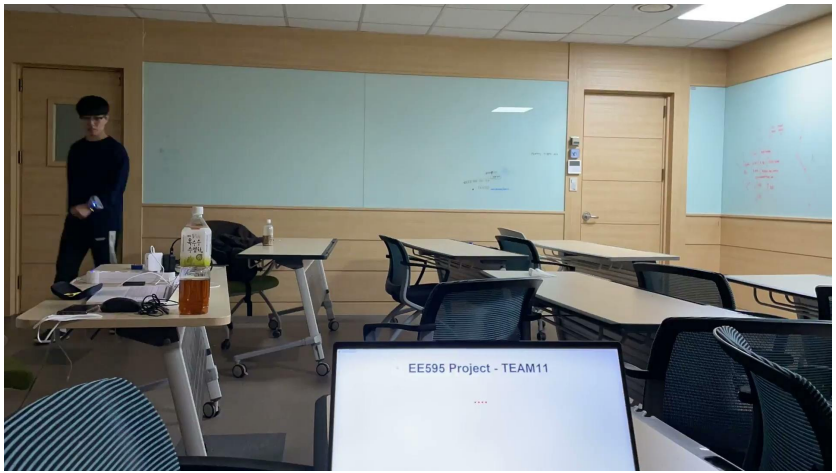
- Accuracy of Voice recognition
  - repeat 50 times (help, other voices)
  - voice : **help**
  - help
    - Precision = 97%
    - Recall = 88%
    - Accuracy = 93%

		answer	
		help	**other voices**
result	**help**	44(TP)	1(FP)
	other voices	6(FN)	49(TN)

Performance

Performance Evaluation

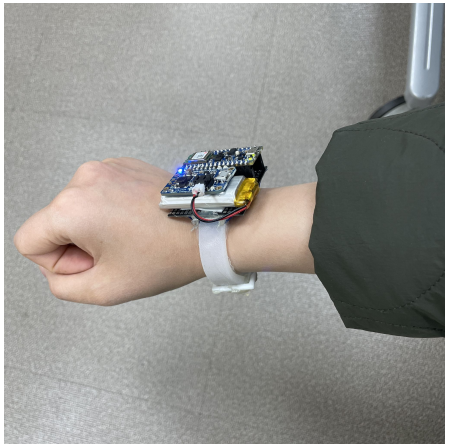
- Testing Scenario : Wear our device for 10 minutes and perform daily activities
  - ex) Talking with friends, Walking



- Result : Emergency situations were not detected in daily life
  - Our Emergency detection algorithm is robust enough in daily life.

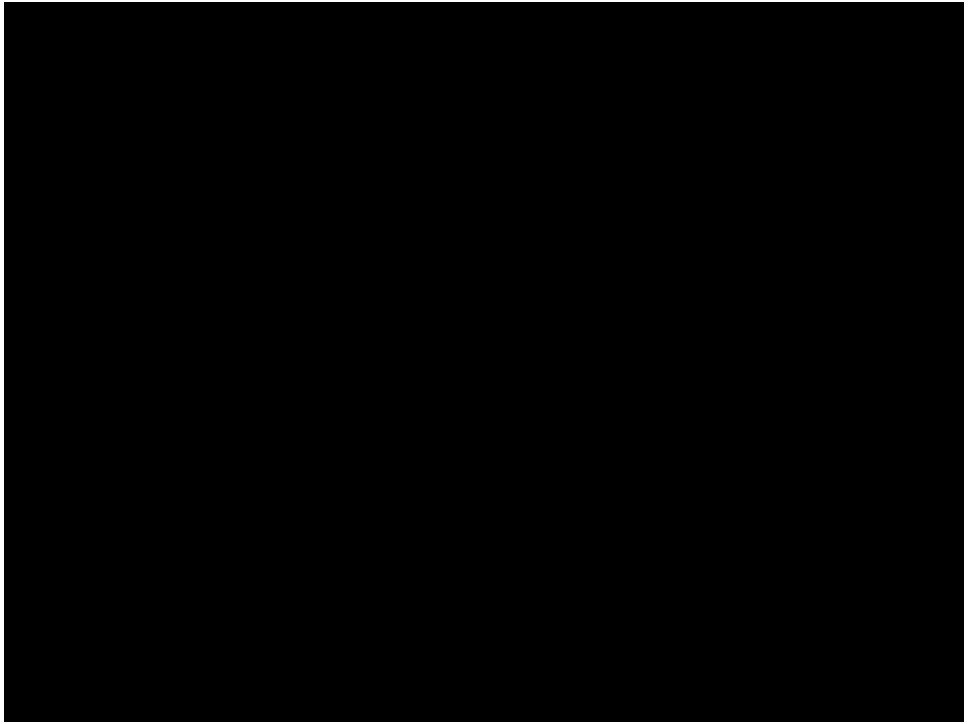
Demo

- We made the device can be worn on the wrist.
  - Consist of Power Boost, Battery and Arduino



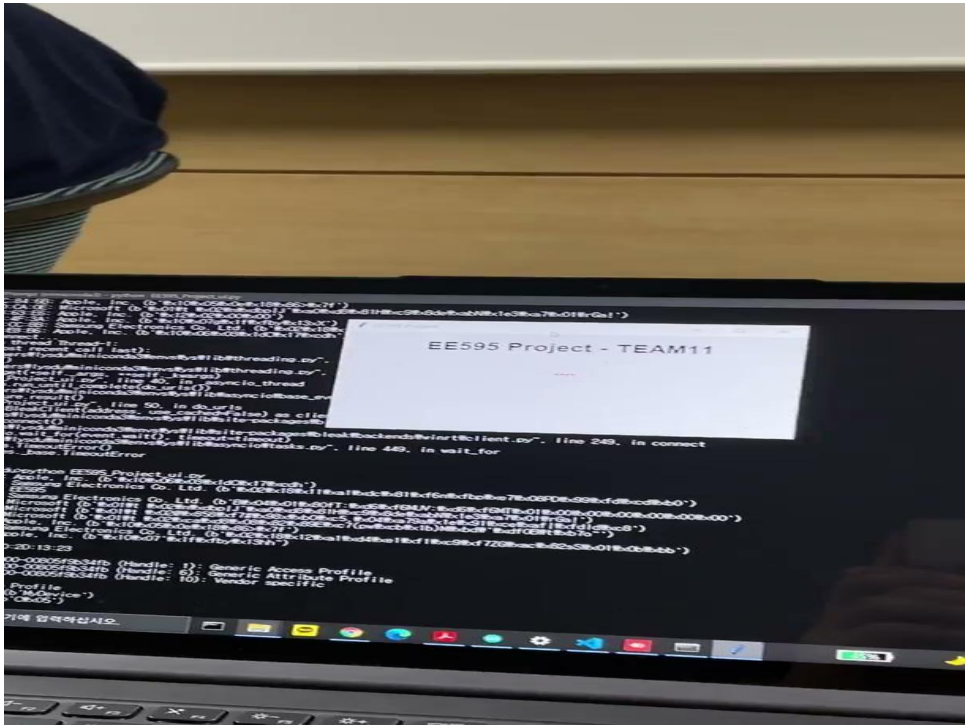
Demo

Situation 1



Demo

Situation 2



Usefulness & Novelty

- Usefulness
  - Can detect **two dangerous situations** successfully.
    - voice recognition accuracy: 93%
    - motion recognition accuracy: 92%
  - it is not detected in a normal situations.
  - **No need** additional equipments.

- Novelty
  - **First service** in dangerous situations, unlike accident situations such as fall detection.

Thank you!

Q&A