Tank Battle

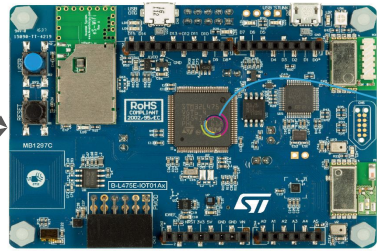
ESLAB Project Report

B08202033 楊智凱
B08202054 李杰銘

Overall Architecture

Button for shooting

Accelero-sensor
Gyro-sensor

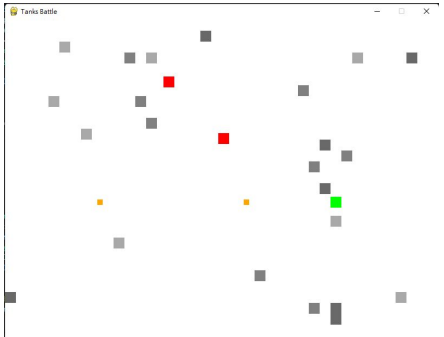


Detect player movement and shooting

WiFi

Multi-threading

Main Program



Goals

Overall goal: Create a tank battle game providing comfortable playing experience

To do so, we need:

1. Processing the data accurately and rapidly enough on STM32
2. Reduce the times of sending data
3. Quick reaction for data received in main program

In addition, some parts of this project can be used in other applications with only slight modification.

Calibrating the sensor

Method:

1. Collecting 3000 data (gyro & accelero) first and calculating the mean of them
2. Store the resulting offset for future need

But we tried:

- Not to calibrate, but collect some data and do DSP filtering before sending action
- Cannot send too frequently since both collecting data and DSP are time-consuming
- In a result, the playing experience got worse.

Detecting shooting

Method:

1. Control by the user button (blue button on our board)
2. Callback function when pressed and released can do this work

Detecting moving

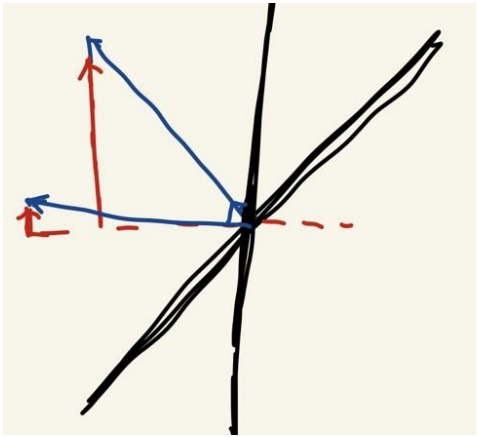
Method:

1. Record the total rotation angle by Riemann Integration of (Gyro - offset)
2. Detect the acceleration data and minus the offset (left & right)
3. One of the above quantities over the threshold -> move
4. Detect the moving every 5ms
5. Different threshold for different operations based on our empirical observations

Detecting moving

Tries:

1. Use change of z-axis acceleration as criterion instead of x-axis or y-axis acceleration -> Poor performance however and hence not adopted



Detecting moving

Tries:

2. Use “Counter” to decide action in order to conquer the problem of “recoiling”

   -> doesn’t have specific criterion applicable for all the scenario

3. Try to “score” each operation, and take action only if the score is high enough

   -> Hard to design an universal scoring policy

Sending data

1. Through wifi socket
2. One operation each time
3. Format: a number ranging from 1 to 5, each representing different action
4. Keep detecting and sending can be load for game main program

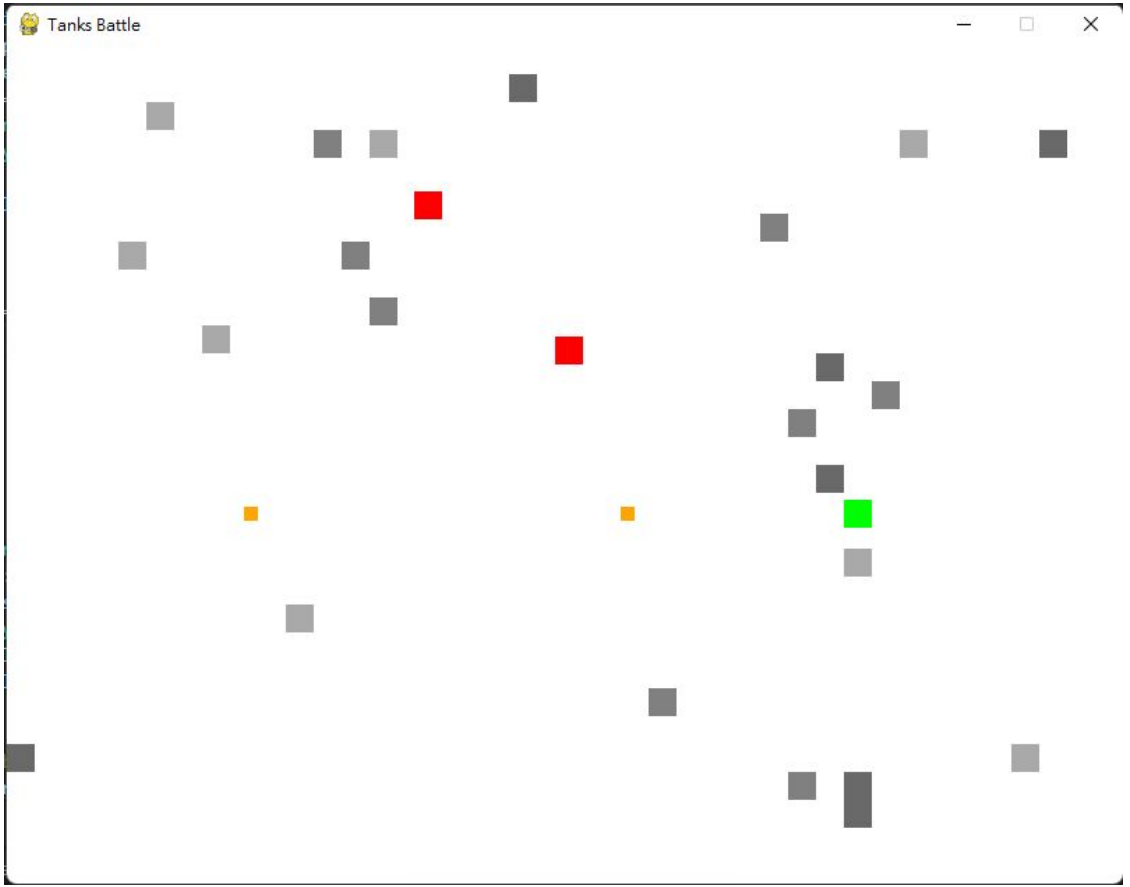
   -> Send operation only if the operation is different from the previous one

   -> This is fine since this is exactly the scenario we play the actual game

5. Not using BLE due to latency and speed concern
6. After the game is over, the socket is disconnected -> reset for second turn

Tank battle game

- Pygame



Main program

- Run game program and socket program simultaneously.
- Multi-threading
- Connect two threads with Event()

- Sensor server receive signal from STM32
- -> Change the state of Event()
- -> Game program detect the state of Event() and take the corresponding action
- After the game is over, restart for second turn

Main program

Adjusting FPS:

1. Too small -> inherent lagging in the game
2. Too high -> cannot match the frequency of STM32 sending data

We set 30 in this project

Demo