# Lab4 report Zhe Liu

## Section 1: Description

This Lab we are required to implement a well-known game “tic tac toe”. There are two players of this game. Computer (using char ‘X’) and user (using char ‘O’). But the game itself is different than this well-known game. Basically, user must use different frequency combination to drive this game. In other word, the location that user puts the “O” char on is based on sound input recorded by microphone.

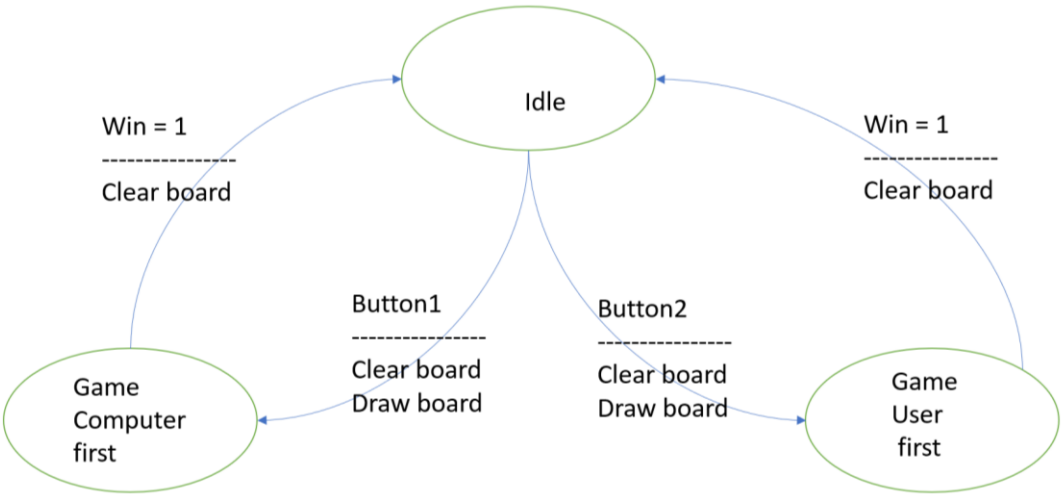
There are two modes in this game. One is let computer play first and the other is let user play first. So, if a column or row or diagonal contains three cells of ‘X’ or ‘O’ marks, then the corresponding player will win this round. If any player win, then they will receive 1 points. Points will display on top of LCD screen. There could be a tie game if there’s no three ‘X’ or ‘O’ marks form a straight row or column or diagonal. Once tied game, no player receives points. There’s different sound with different LED blinking corresponding to different wins.

## Section 2: FSM design

void ProcessStep(int S1d, int S2d, int prev_S1d, int prev_S2d, int sec, int ms50, int sec3, int sec2);

typedef enum {idle, gameS1, gameS2} state_t;

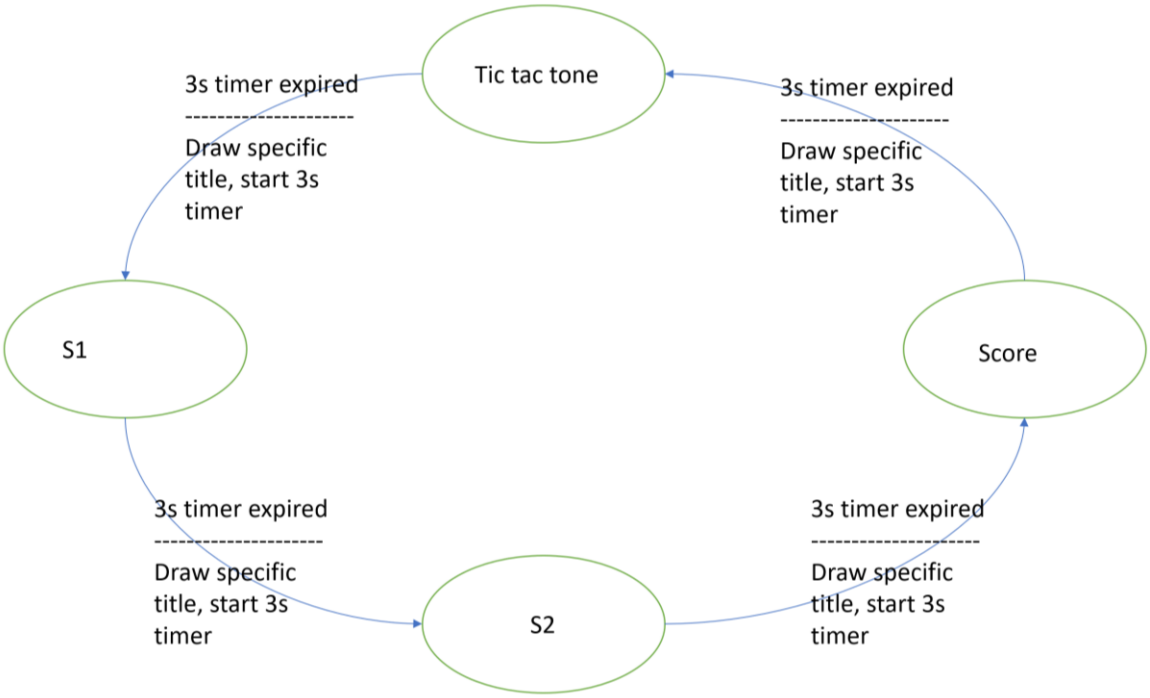
This is the main FSM. It uses to switch between gameS1 and gameS2 and screen saver that is idle. Game S1 is let computer play and Game S2 is let user play. It uses button 1 and button 2 to select Game mode.



void drawTitle(int sec3, gamestate_t G);

typedef enum { title _title, title_s1, title_s2, title_score} title_state_t;

This is the title FSM. It used to switch the title that displayed on the top of LCD. When 3 sec timer expired, it will switch to another state.




state_t ProcessIdle(int sec, int ms50, gamestate_t *G)

typedef enum{

processidle_idle,

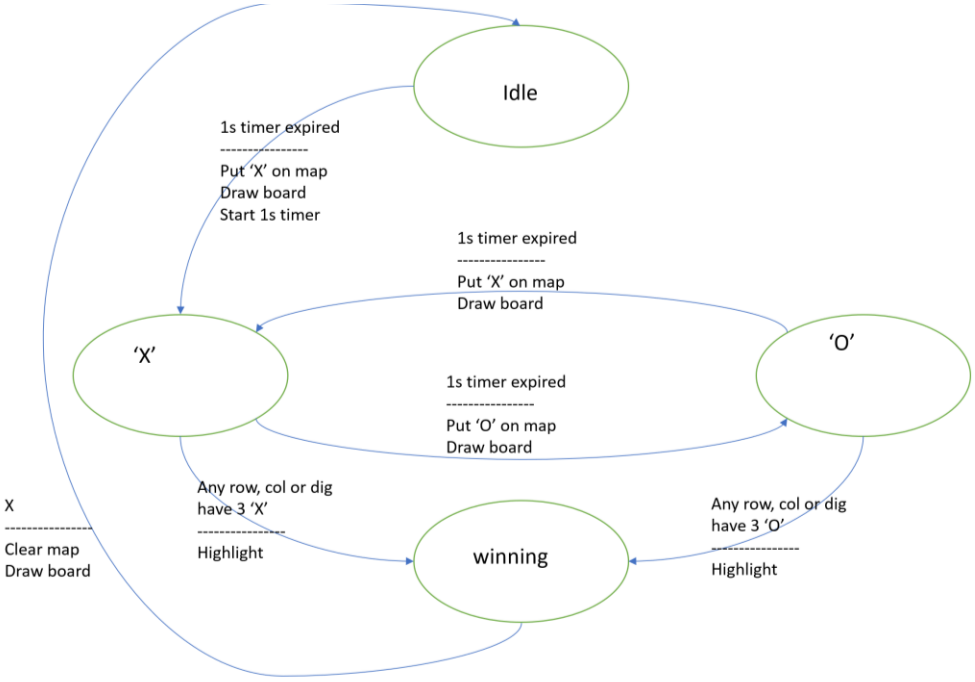
processidle_playingcircle,

processidle_playingcross,

processidle_winning

} processidle_state_t;

This FSM is for screen saver. Basically, play this game by computer itself. However, it will determine which char win the game. Then highlight it. Or, it could tie the game.

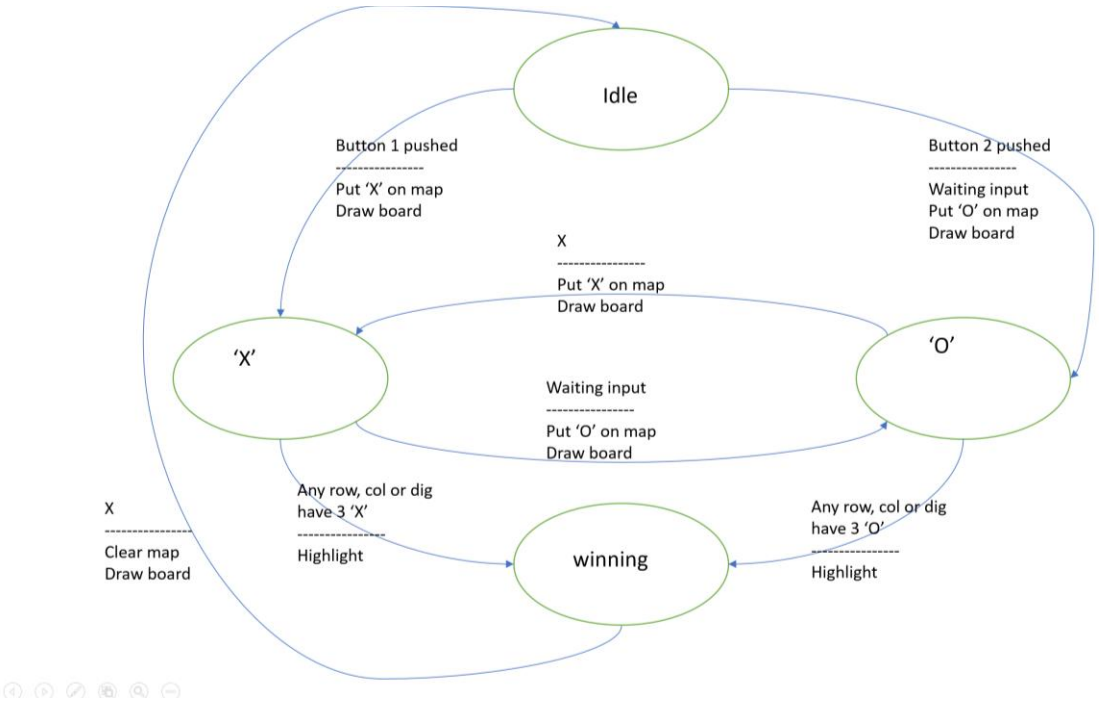



void ProcessS1(gamestate_t *G, int sec2);

void ProcessS2(gamestate_t *G, int sec2);

typedef enum{waiting, playingcircle, playingcross, winning, waiting2, waiting3}state_t1;

These FSMs are for game mode1 and game mode2. They are almost like screen saver. Basically, play this game by computer and user. The only difference between mode 1 and mode 2 is who play first. However, it will determine which char win the game. Then highlight it. Or, it could tie the game.




Event sequence:

void ProcessStep(int S1d, int S2d, int prev_S1d, int prev_S2d, int sec, int ms50, int sec3, int sec2);// main FSM

case idle:

state_t ProcessIdle(int sec, int ms50, gamestate_t *G); // for screen save

void drawTitle(int sec3, gamestate_t G); // draw title on the top of screen

case gameS1:

void ProcessS1(gamestate_t *G, int sec2); // this is game mode 1

case gameS2:

void ProcessS2(gamestate_t *G, int sec2); // this is game mode 2

## Section 3: HAL design

HAL list:

**For button:** button.c, button.h

void InitButtonS1();

int ButtonS1Pressed();

void InitButtonS2();

int ButtonS2Pressed();

This is button HAL used for all button related functions.

**For display:** display.c, display.h

```
/* palette */
#define BACKGROUNDCOLOR GRAPHICS_COLOR_PURPLE
#define LINECOLOR       GRAPHICS_COLOR_WHITE
#define FOREGROUNDCOLOR GRAPHICS_COLOR_CYAN
#define EMPHASISCOLOR   GRAPHICS_COLOR_YELLOW
#define XCOLOR GRAPHICS_COLOR_DEEP_PINK
#define OCOLOR GRAPHICS_COLOR_GAINSBORO
//above are all color setup
typedef struct {
   tcellstate map[9];
   int computerscore;
   int humanscore;
} gamestate_t;
```

void InitDisplay();

void DrawTime   (unsigned minutes, unsigned seconds);

void DrawMessage(char *s, uint32_t color);

void DrawScore  (int computerscore, int humanscore, uint32_t color);

void DrawBoard  (tcellstate map[9]);

void DrawBoardIdle  (tcellstate map[9]); //since idle state don’t need to change color, therefore this function only used for idle state

void DrawWinner (tcellstate map[9],int winner, uint32_t color);

void DisplayMaxMin(unsigned max);

This is display HAL for all display including message, grid, draw board etc.

**For encoding:** dtmf.c, dtmf.h

```
#include "goertzel.h"
#include "display.h"
#include "maplogic.h"

void     DTMFAddSample(unsigned x);
void     DTMFReset();
unsigned DTMFPower();
int      DTMFCheck();
void drawCir(gamestate_t *G,int winner);
```

This HAL is for encoding. Basically, decide which map place that should be taken by ‘O’.

**For goertzel filter:** goertzel.c goertzel.h

```
#define PTHRESHOLD 1000
#define FRACBITS 8
#define FIXMUL(A, B) ((A * B) >> FRACBITS)

typedef struct {
   int coef;
   int s2, s1;
} Gtap;

int  ScaleSample   (unsigned s);
void SampleGoertzel(Gtap *t, unsigned x);
void ResetGoertzel (Gtap *t);
int  PowerGoertzel (Gtap *t);
```

This is filter HAL. Use fix point arithmetic for future use.

**For hardware timer:** hwtimer.h, hwtimer.c

#define SYSTEMCLOCK 16000000

void InitTimer();

Hardware timer HAL. For timer initialize and all related functions.

**For LED:** led.c led.h

void InitLEDs();

// 3 functions for LED on booster

void Toggle_Booster_LED();

void TurnON_Booster_LED();

void TurnOFF_Booster_LED();

void redLED();

void greenLED();

void blueLED();

void yellowLED();

void violetLED();

void cyanLED();

void whiteLED();

LED HAL used for all LED. LED initialize, toggle and different color.

**For map logic:** maplogic.h maplogic.c

typedef enum {empty, cross, circle} tcellstate;

```
// This function returns true when X wins
int      CrossWins(tcellstate map[9]);

// This function returns true when O wins
int      CircleWins(tcellstate map[9]);

// This function resets map sate to empty
void     ClearMap(tcellstate map[9]);

// This function fills all empty cells with 'X'
void     AbortMap(tcellstate map[9]);

void     CheatMap(tcellstate map[9]);

// Adds a symbol v in a random empty location
int      RandomAdd(tcellstate map[9], tcellstate v);

// This function returns true of neither O nor X wins and no more moves are possible
int      Tie(tcellstate map[9]);
```

This is logic HAL for check who wins. By check, I mean to find if there's any col or row or diagonal that forms a straight line by either 'X' or 'O'. Then highlight this row/col/diagonal.

**For microphone:** microphone.h microphone.c

```
void InitMicrophone();
unsigned GetSampleMicrophone();
```

This is microphone HAL. It's for all microphone related functions.

**For sound:** sound.c sound.h

```
typedef enum
{
    note_silent,
    note_c4,
    note_d4,
    note_e4,
    note_f4,
    note_g4,
    note_a4,
    note_b4,
    note_c5,
    note_d5,
    note_e5,
    note_f5,
    note_fs5,
    note_g5,
    note_a5,
    note_b5,
    note_c6
} tnote;

void InitSound();
void PlaySound(tnote n, unsigned ms);
void playSoundCircle();
void playSoundCross();
void playSoundTie();
void computerSound(int i);
```

This is sound HAL. Different player wins will play different sound. So all sound setups are in this HAL.

**For software timer:** swtimer.h, swtimer.c

```
typedef struct {
   uint32_t hwtimer;  // hardware timer used as basis for this software timer
   uint32_t period;   // period of the software timer
   uint32_t bound;    // next expiration time for software timer
   int      expired;
} tSWTimer;


void InitSWTimer(tSWTimer *T,
                 uint32_t  hwtimer,
                 uint32_t  period);
void StartSWTimer(tSWTimer *T);
int SWTimerExpired(tSWTimer *T);
int SWTimerOneShotExpired(tSWTimer *T);
```

Software timer HAL. For timer initialize and all related functions.

## Section 4: Program Structure

Once entered this game, first enter to idle mode. Idle mode is basically computer playing with computer. Title will change once 3 seconds. Mode 1 and mode 2 are in parallel structure. To enter mode 1, user must push button 1. Same as mode 1, to enter mode 2, user must push button 2.

After push button 1, the program will first clear all map and re-draw the board. Then the computer will take the first step by putting ‘X’ on a random place first (1). After completing this process, game will wait user to give a command. This command is generated by DTMF signal. Once received this signal, game will put ‘O’ on the place that user want (2). Then repeat step (1) and (2) until find the winner or tied. In each process, game will check if there’s a winner. If all 9 places have been taken and there’s no winner, then the game is tied. Once decide the winner, draw a title “who win”, play a specific sound and blink LED. Then back to Idle state.

I have 3 Bonus parts in the LAB.

1. Blink LED when winner decided. Three different LED blink format based on computer wins or user wins or tied game.
2. Different color ‘X’ and ‘O’ symbol. I set ‘X’ as color hot pink and ‘O’ as color gray. But this color change won’t affect idle mode.
3. Meter on the bottom. When microphone get input voice, it will return volume on the bottom of screen. But there’s still bugs in this meter.
4. Cheat mode. When push button 2 during game, then user will automatically win this game.