Gesture Hand Game

Siji Oluwadara
January 8th, 2023

Application description

A fun game that helps with memory retention and ear training. The device will play a sequence of sounds that the user will respond to with gestures that correspond to the sounds. There are 3 levels. The higher the level the more notes the user must remember.

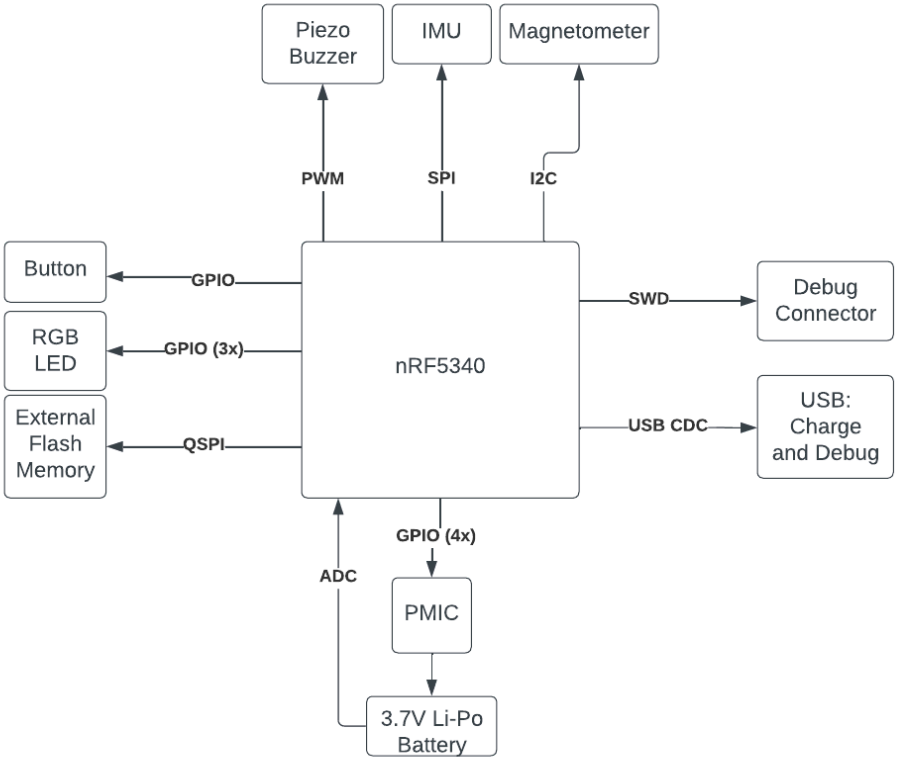
Hardware description

The processor on the thingy53 is a dual core Arm Cortex-M33. One core is the application core and the other is a network core (for wireless connectivity). The application core has 1MB of Flash and 512 kB of RAM. The network core has 256kB of Flash and 64kB of RAM. There is an external Flash device with 64MB of memory.

BOM

Component	Part Number	Price (1 unit)	Link
Development board	Thingy53	$60.03	https://www.digikey.com/en/products/detail/nordic-semiconductor-asa/THINGY53/15211700
MCU	nRF5340	$9.04	https://www.digikey.com/en/products/detail/nordic-semiconductor-asa/NRF5340-CLAA-R/14323741
6 Axis Accelerometer + Gyroscope	BMI270	$5.88	https://www.digikey.com/en/products/detail/bosch-sensortec/BMI270/9974486
Buzzer	-	-	-

Block Diagram



The hardware components on the thingy53 that will be used for the game:

- The RGB LED that will display that the device is on and other status signals
- The buzzer will output the sequence of sounds
- The 6 axis IMU (accelerometer and gyroscope), BMI270, which will be used for identifying gestures
- A button will be used to reset the game
- The console via USB will be used for debugging
- The SWD interface will be used for programming

Pin Table

Pin Name	Pin Number	Hardware Interface
RED	P1.08	Red LED
GREEN	P1.06	Green LED

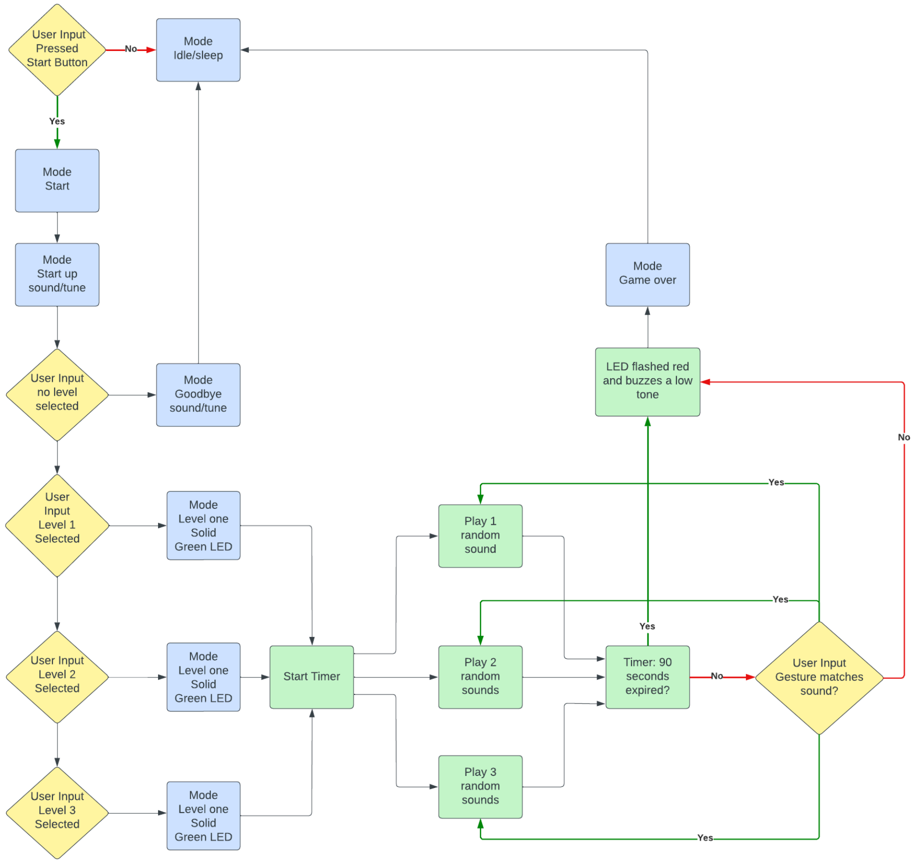
BLUE	P1.07	Blue LED
BUZZER	P1.15	Buzzer
INT1	P0.23	BMI270 interrupt
CS	P1.04	BMI270 chip select
SCK	P0.29	BMI270 SPI Clock
MOSI	P0.28	BMI270 SPI MOSI
MISO	P0.26	BMI270 SPI MISO
BUTTON1	P1.14	Button

Modules

- State Machine.c - is the logic and contoller of the game
- Imu.c - initializes the IMU, collects the accelerometer and gyroscope data and determines the orientation of the device
- Led.c - initializes the LED GPIO ins and selects the different LED states
- Buzzer.c - initializes the buzzer and selects the different preset songs that are saved in struct arrays. It also includes a struct array of the device sides with their corresponding tone.
- Button.c - initialize the Button GPIO. Sets up the interrupt, initializes the k_timer and k_work so that that game can start or so a level can be selected.

State machine

The major change to the state machine was removing the 90 second timer and instead distinguishing between levels by the number of notes the player needs to get right in a sequence. This decision was made because of the complexity of having multiple threads.


User Input Pressed Start Button
No
Mode Idle/sleep
Yes
Mode Start
Mode Start up sound/tune
User Input no level selected
Mode Goodbye sound/tune
User Input Level 1 Selected
Mode Level one Solid Green LED
User Input Level 2 Selected
Mode Level one Solid Green LED
User Input Level 3 Selected
Mode Level one Solid Green LED
Start Timer
Play 1 random sound
Play 2 random sounds
Play 3 random sounds
Timer: 90 seconds expired?
Yes
No
User Input Gesture matches sound?
Yes
Yes
Yes
No
LED flashed red and buzzes a low tone
Mode Game over

Link to table above: https://docs.google.com/spreadsheets/d/1GFFZW3i4PAbwur2Zs7Fo1XQzepI3AlYUf7l8aIuqGmg/edit?usp=sharing

State	Action	Animation	Time out selection	Button pressed once time	Button pressed twice times	Button pressed three times	Gesture matched tone sequence
START	Play start up tune	Start up intro	IDLE/SLEEP_ MODE	LEVEL1_ MODE	LEVEL2 _MODE	LEVEL3 _MODE	-
NEXT_LEVEL	Choose level mode based on number of button presses	Blinking LED					

LEVEL1_MODE	One button press is level 1. Start Timer	Solid Green LED on	LED flashes red and buzzes a low tone. Go to GAME_OVER_ MODE	-	-	-	Play one random tone
LEVEL2_MODE	Two button presses are level 2. Start Timer	Solid Teal (green and blue) LED on	LED flashes red and buzzes a low tone. Go to GAME_OVER_ MODE	-	-	-	Play two random tone
LEVEL3_MODE	Three button presses are level 3. Start Timer	Solid blue LED will be on	LED flashes red and buzzes a low tone. Go to GAME_OVER_ MODE	-	-	-	Play third random tone
GAME_OVER	-	Flash the red LED and buzz a low tone	-	-	-	-	-

Overall

CLI commands (To do)

- I2C scan
- GPIO scan
- Firmware information: version, mcu p/n

Software tools

- VSCode
- VMware for Linux VM
- Zephyr RTOS: board configuration
- Zephyr Toolchain: CMAKE and Ninja and GCC Compiler to compile and build

Challenges:

1. Threads. I spent hours debugging because I did not fully understand threads and then I didn’t understand threads in zephyr.

2. I could not debug the device via USB so I had to debug using printk statements.
3. I had to program using nrfConnect programmer. It took a very long time (~2-3 mins) to program the device. In addition, I had to press a small button to put it into bootloader mode. This proved difficult at times because the button was small.