Architetture dei Sistemi
Di Elaborazione

Delivery date:
9th Gennaio 2023

Extra point #1
Max 2 points

Expected delivery of extapoint1.zip must include:

- zipped project folder
- this document compiled in pdf
- A 4 minutes video with audio (.mp4 or .avi) explaining how the project works; the video has to show a software debug session with all significant peripheral windows opened; the audio must be a voice recording of you describing (in Italian or English) the behavior of the running system.

Purpose of Part 1: to acquire full confidence in the usage of the KEIL **software debug** environment to emulate the behaviour of the LPC1768 and the LANDTIGER Board.

This part is evaluated to assign a maximum of 2 extra-points for qualified students taking the exam with a mark >= 18

Make a Tamagotchi!

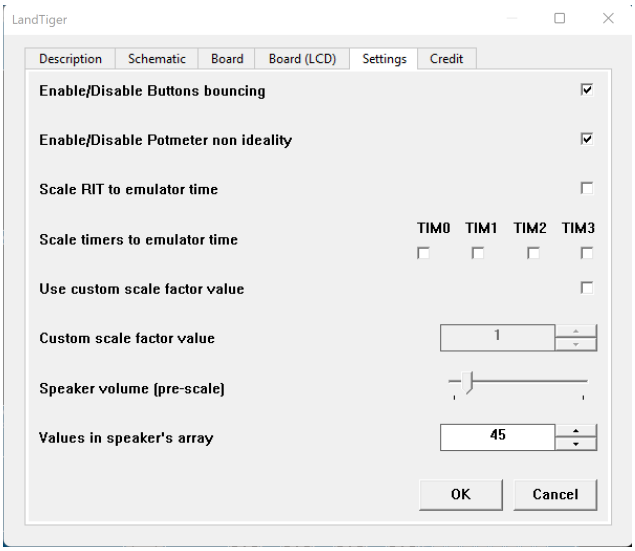
Tamagotchi is a famous toy from the end of the '90s and the beginning of the new millennium (but they still make them with genetics, NFC, and Bluetooth!). The goal of the game was to take care of a virtual pet, taking care of its basic needs, such as hunger and happiness. Of course, in the end, many kids just had tons of them on their consciences.

In Keil µVision, use the LANDTIGER **emulator** (or the actual board if you are able to have one) for implementing a basic Tamagotchi!

Please deliver a zip folder with all the files of your project. At the end of the Keil project folder, please add "EMULATOR" if you developed it using the emulator or "BOARD" if you had the opportunity of developing the project on the actual board.

Example: extrapoint1_emulator.zip **or** extrapoint1_board.zip

Please attach any useful comments and your emulator configurations (substitute the image below). Your project will be evaluated according to your specification!!



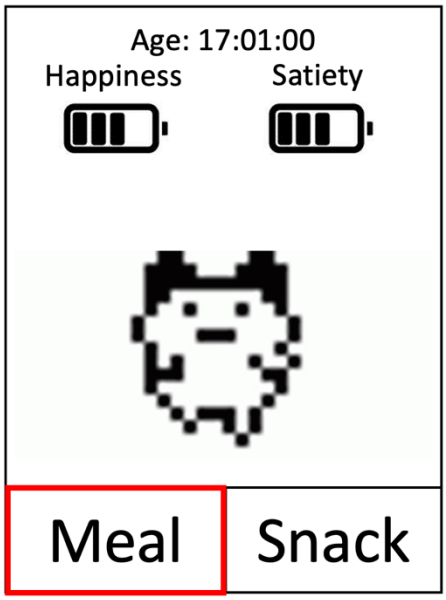
Additional Comments (C-variable/defines defined in your code, e.g., scaling factors) :
Data l'impossibilità di scalare il RIT dato da un crash della intera interfaccia di Keil al click destro sulla scheda in debug (su macchina virtuale) è stato usato Timer2 al fine di implementare il meccanismo di polling ogni 50 ms. Costanti da modificare nel caso il gioco risultasse troppo lento: • #define LOSING_SATIETY_PER_SECOND 1 → tasso di perdita sazietà al secondo • #define LOSING_HAPPINESS_PER_SECOND → 1 tasso di perdita felicità al secondo typedef enum { MEAL, SNACK } selection; → per tener traccia della selezione corrente typedef enum {ALIVE, END, EATING} status; → per tener traccia dello stato corrente Età calcolata sulla base del Timer Counter di Timer1 e la frequenza di clock XTAL.

Specification 1)

Use the display library to create and show your own Tamagotchi. Be creative! A basic "animation" is also required for this point. You are free to decide what kind of animation to create and how complex you want to make it but remember that there is a limit on the dimension of the code that you can compile and device performances, so just try and be ready to simplify it if needed.

Requirement 1: You are required to display the Tamagotchi with the shape of your choice in the middle of the LCD.
The animation has a refresh rate of 1 second.

Make it alive!

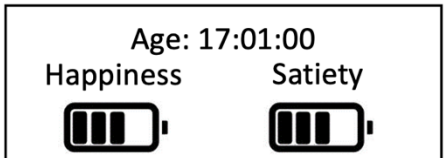


Specification 2)

Create the logic of your virtual pet.

At the top of the screen, the STATUS of your virtual pet is shown. The status view must always be visible at the top of the screen, together with Tamagotchi. There are three information in this status view:

a) The "Age" of your Tamagotchi (in hours:minutes:seconds);
b) The "Happiness" of your Tamagotchi (on a scale of your choice);
c) The "Satiety" of your Tamagotchi (again on a scale of your choice).



Create a FOOD MENU on the bottom of the screen. The food menu must always be visible at the bottom of the screen, together with Tamagotchi.
Your icons may be different or simplified text abbreviations. Note that the RED square is the currently selected menu. The option in the food menu should be:

a) "Meal" to satisfy the hungry level of your Tamagotchi;
b) "Snack" to satisfy the happiness level of your Tamagotchi.

Selecting a food or a snack triggers an animation with a sketch of the selected food (it may be a simple circle or square, just differentiate the snack from the meal) and the Tamagotchi "eating" it. It's not necessary to make a complex animation, but the Tamagotchi must move toward the food/snack (appearing in a fixed position on the screen) for eating.

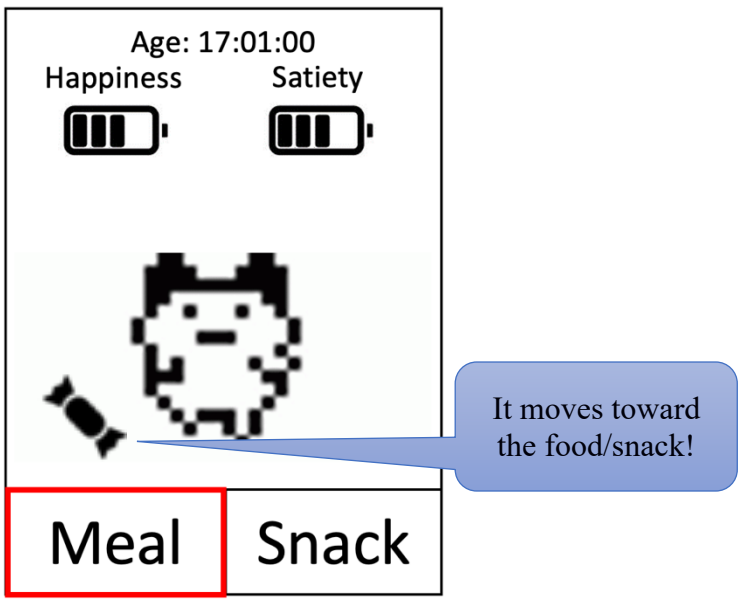
The eating must last for at least 1 second.
Food selection during eating is disabled.

- Eating a snack increases the happiness of your Tamagotchi by one bar (value of your scale).
- Eating a meal increase the Satiety of your Tamagotchi by one bar (value of your scale).

Requirement 2: You are required to develop the Food/Status menu.
Note that images are only for explanation purposes.

Note that "Happiness" and "Satiety" decrease every 5 seconds of one bar (value of your scale).

The **JOYSTICK** is used to select the food options (**LEFT/RIGHT** for choosing the desired food and **SELECT** to choose it).



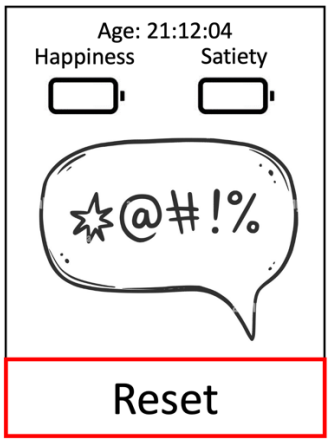
The eating must last for at least 1 second.
Food selection during eating is disabled.

Your Tamagotchi runs away!

Specification 2)

Your Tamagotchi should leave you if you don't take care of it. If you reach the end of the Happiness or Satiety scale, the runaway sequence is triggered.
You should have a very basic animation of the Tamagotchi leaving the screen. After it's run away:

a. A message or icon should be placed in the screen to signal that your Tamagotchi left you.
b. The age counter stops counting the age of your Tamagotchi.
c. A reset button replaces the food options.



By clicking the reset button, a complete reset of your Tamagotchi is triggered, with a new Tamagotchi appearing on your screen, with **full Happiness and Satiety and with 00:00:00 as age.**

Requirements 3:
You are required to develop the runaway sequence and the reset functionality.
Note that images are only for explanation purposes.

For the Reset functionality, you simply must press the joystick (**SELECT**).