Software Formalization

Year: 2023 Semester: Spring Team: 8 Project: Engineer’s Chess

Creation Date: 02/13/2023 Last Modified: 02/25/2023

Author: Bazim Azeem Email: bazeem@purdue.edu

Assignment Evaluation:

| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| --- | --- | --- | --- | --- |
| **Assignment-Specific Items** | | | | |
| **Third Party Software** |  | x2 |  |  |
| **Description of Components** |  | X3 |  |  |
| **Testing Plan** |  | x3 |  |  |
| **Software Component Diagram** |  | x4 |  |  |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** |  | x2 |  |  |
| **Formatting and Citations** |  | x1 |  |  |
| **Figures and Graphs** |  | x2 |  |  |
| **Technical Writing Style** |  | x3 |  |  |
| **Total Score** |  | | |  |

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

1.0 Utilization of Third Party Software

We do not plan on using any third party libraries or software on our microcontroller apart from the standard header files provided in the STM32 Workbench.

We will, however, use a library on the Jetson Nanao to implement the processing of the analog voice input into a string of commands. As there are many libraries available for this purpose, we have narrowed down our options to two:

| **Library** | **License** | **Description** |
| --- | --- | --- |
| Picovoice Platform  NVIDIA Jetson  (our most likely candidate) | Picovoice | “Picovoice is the end-to-end platform for building voice products on your terms. Unlike Alexa and Google services, Picovoice runs entirely on-device while being more accurate. Using Picovoice, one can infer a user’s intent from a naturally spoken utterance such as:  *Hey Edison, set the lights in the living room to blue.*  Picovoice detects the occurrence of the custom wake word (Hey Edison), and then extracts the intent from the follow-on spoken command.” [1]  This software has a free version which we can purchase and provides the basic functionality we require. |
| PocketSphinx | MIT | “This is PocketSphinx, one of Carnegie Mellon University's open source large vocabulary, speaker-independent continuous speech recognition engines.” [2] |

In addition, we plan on using the python-chess [3] library to help implement the chess game logic on the Jetson Nano. This library includes a variety of objects and functions which greatly facilitate the implementation of a chess game. The most relevant object for our use is Board, which keeps track of all the pieces and their positions, legal moves in the current position, the current player, and if stop conditions are met. This software may be used under the GNU General Public License v3.0.

2.0 Description of Software Components

There are three major software components in this project: the game logic, the voice processing, and the basic I/O.

* The game logic keeps track of the board state, the players, and the timers. The game will be a loop that continues as long as no stop condition (checkmate, stalemate, insufficient material, time out) is met. In each iteration of the loop, the current player is switched, and the current move is received, the board is updated, and appropriate feedback is sent. It sends the board and feedback to the microcontroller via UART.
* The voice processing component handles receiving the voice input from the microphone and processing that into text. This text is then converted into a chess move in the Universal Chess Interface format.
* The basic I/O component will be a collection of functions run on the microcontroller which handles receiving chess move from Jetson Nano, receiving button input using GPIO, displaying the board on the LED matrix via GPIO, and displaying the timers and feedback via SPI on the OLED display. This component also involves the initialization of the appropriate peripherals and formatting the input and output.

The main functions and parts of these three major components are outlined in the Appendix.

3.0 Testing Plan

3.1 Game Logic

The game logic is the most important part of the project. This, however, will be quite easy to verify. We will test this component by first running the code on a computer, using the command line to input moves and printing the output board to the command line as well. We will play a few games, aiming to end the game using the different stop conditions outlined earlier. This will give us a good idea of the functionality of this component. Once the basic I/O component is working, we can test these components concurrently, using the I/O parts to display the game and receive input.

3.2 Basic I/O

The basic I/O component is the second most important to this project. This will be tested on a piece-by-piece basis as each input/output part is performing a different task.

* Receiving chess move from Jetson Nano: We can test this by using an A2D to look at the data transmitted. We could also print the received input to the console or the feedback display using the microcontroller.
* Receiving button input: This will be trivially tested by lighting an LED when the button is pressed.
* Outputting board to LED Matrix: We will test this part by attempting to send some form of test output to ensure that the GPIO bit-banging process is functional. Then, we can try to display a sample chess board along with the pieces.
* Outputting feedback and timers to OLED display: We will test this part by attempting to send some form of test output to ensure that the SPI communication is functional. Then, we will try to print text in the desired position. Finally, we can attempt to print the formatted timers.

3.3 Jetson Nano

The Jetson Nano (voice processing) component is the least important for this project because we can have a functioning chess game without it. It will have 3 main subcomponents that require testing:

* Receiving voice input from microphone: We can test this by playing back the audio input from the ADC and listening for its clarity.
* Converting voice into text output: We can manually test this by providing voice input into the model and examining the output for correctness. With this being a third party library, the chances of errors are lower as it will already have been extensively tested by the creators.
* Converting text into a chess move: We can test this by giving the function a number of different inputs and observing the behavior on each of these inputs.

4.0 Sources Cited:

[1] Picovoice. (n.d.). *Picovoice Platform* [Online]. Available:

<https://picovoice.ai/docs/picovoice/>

[2] PocketSphinx. (n.d.). *PocketSphinx 5.0.0* [Online]. Available:

<https://github.com/cmusphinx/pocketsphinx>

[3] Niklas Fiekas. (2022). python-chess: a chess library for Python [Online]. Available:

<https://python-chess.readthedocs.io/en/latest/index.html>

Appendix 1: Software Component Diagram

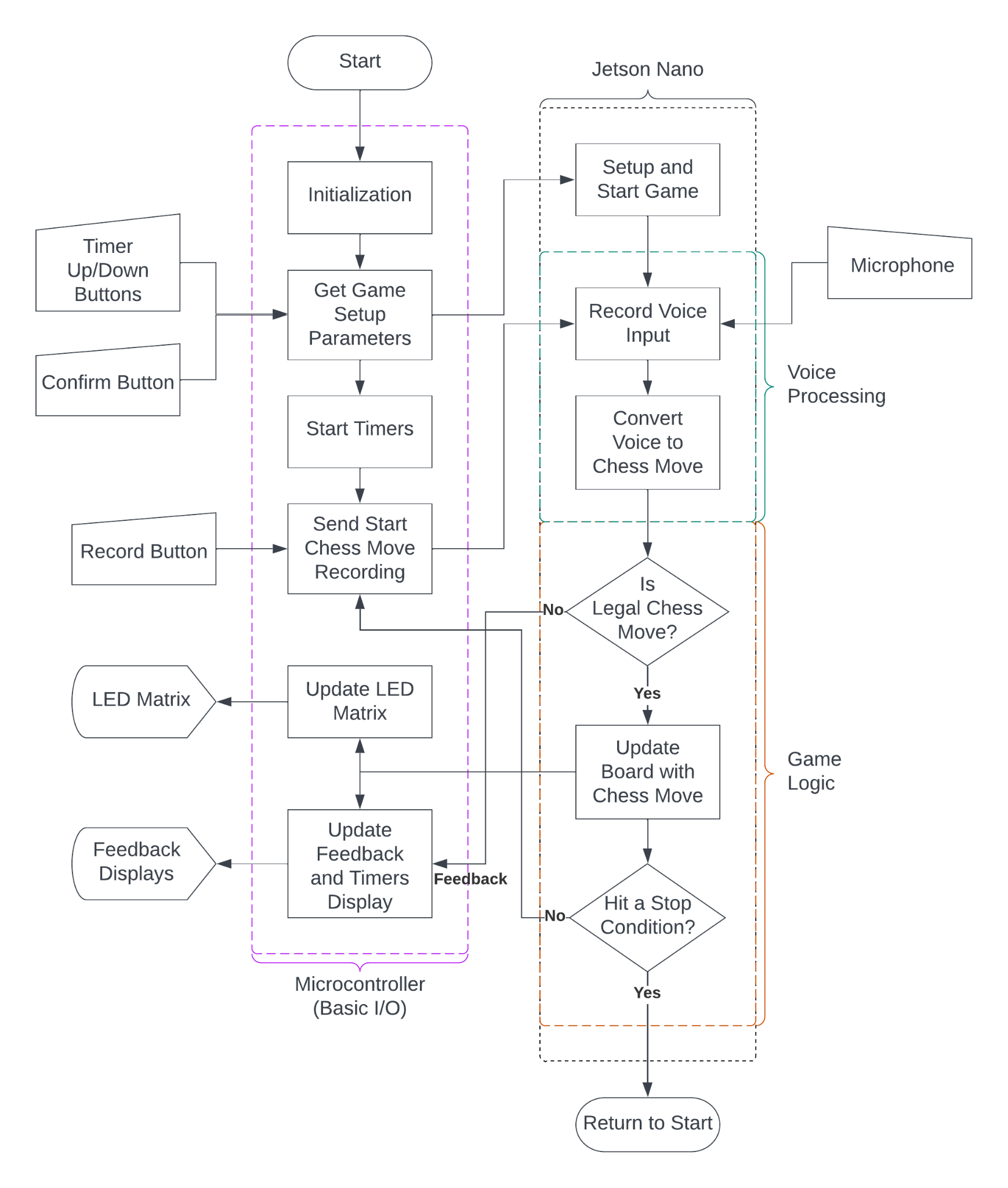
**

Figure 1. Basic code structure

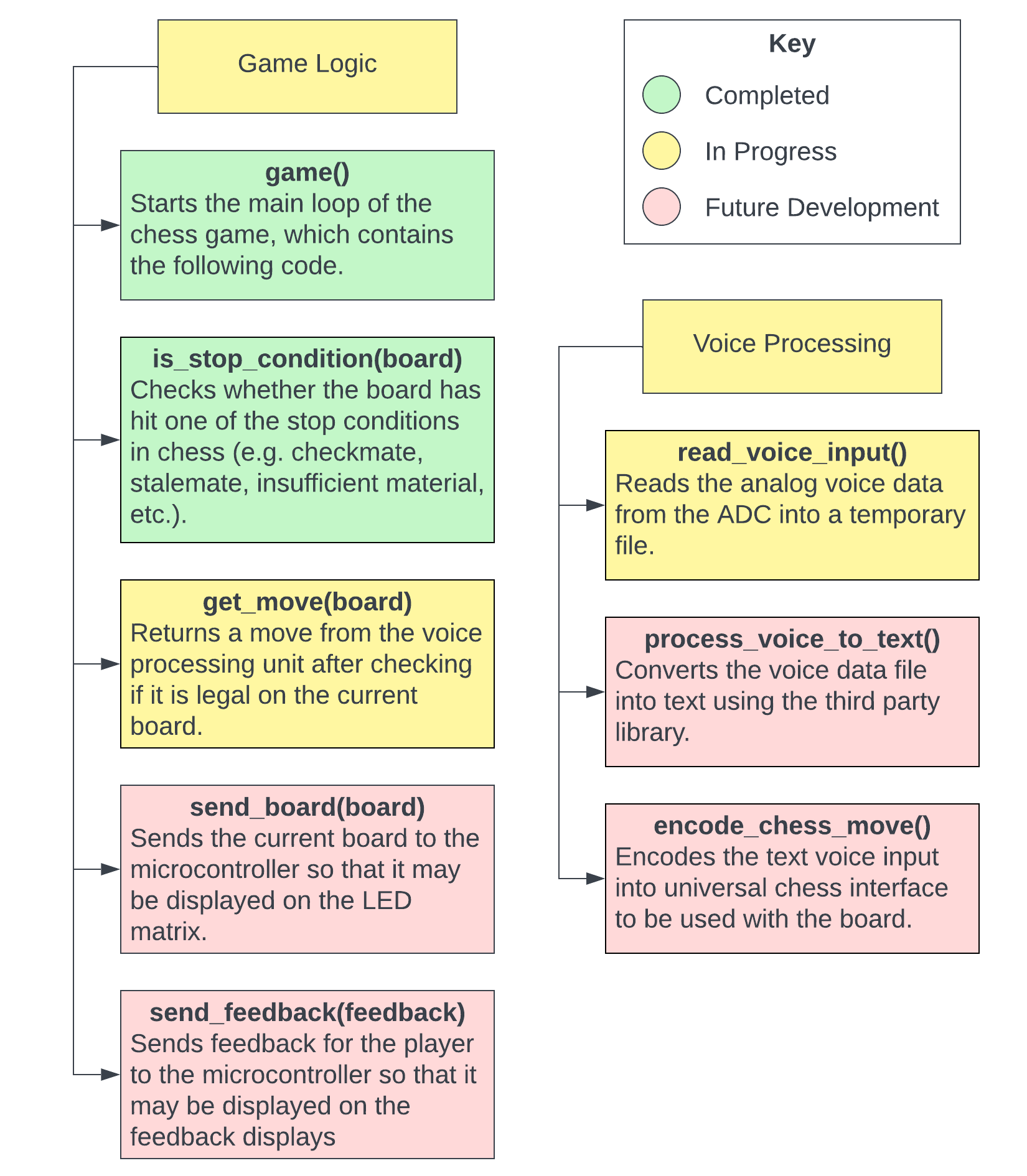
**

Figure 2. Descriptive diagrams of Jetson Nano and Game Logic components.

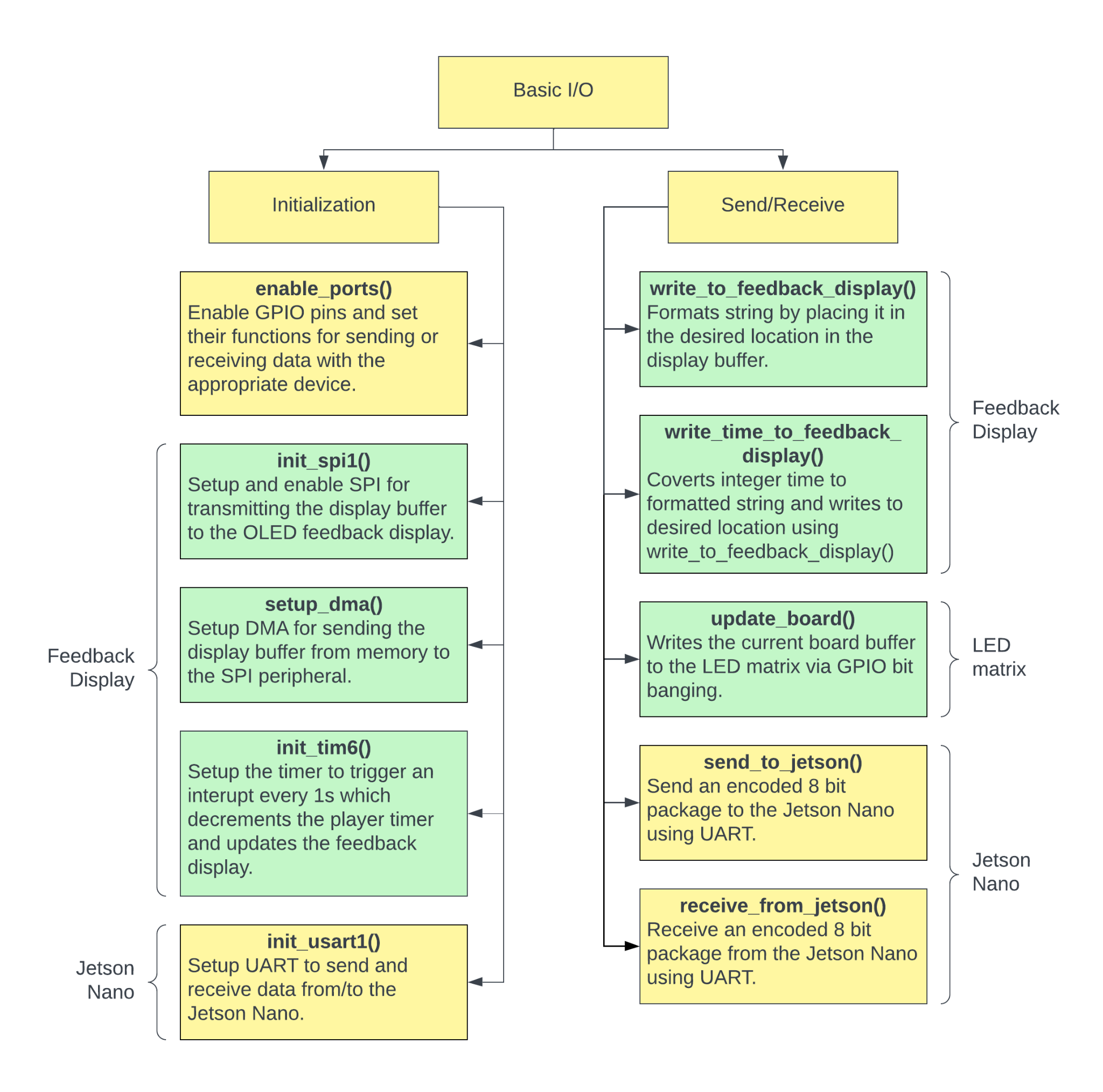
**

Figure 3. Descriptive diagram of the Basic I/O component.