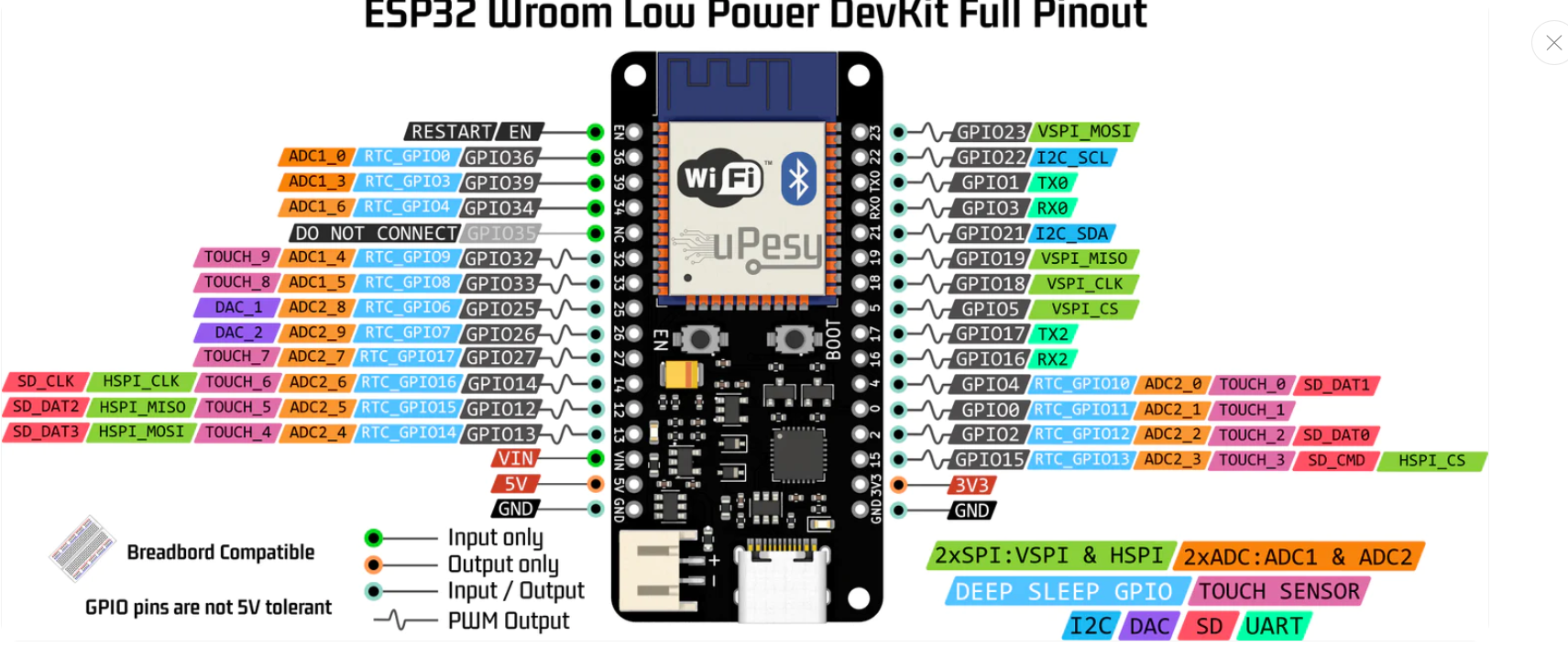
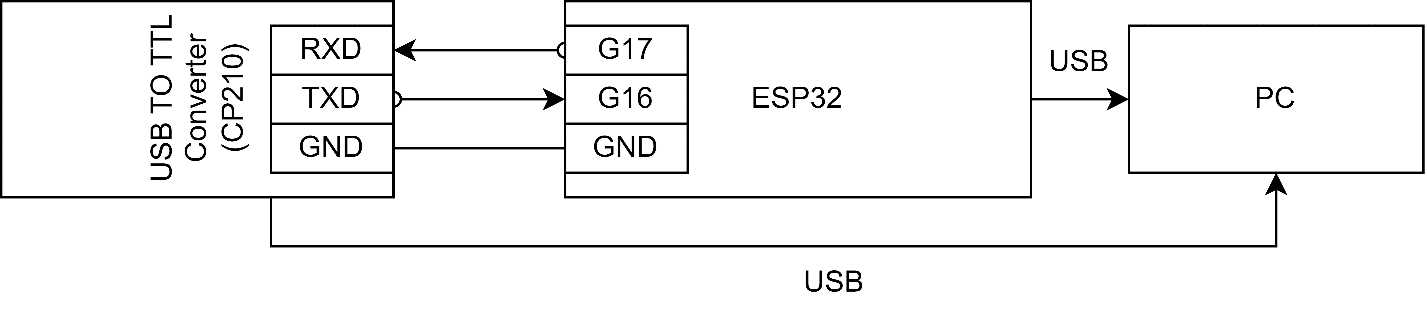
**Hardware Setup**

In this project, it is supposed to have the communication with ESP serial, that is why we need to have USB to TTL dongle. In ESP, we have GPIO16 and GPIO17 assigned as the RX2 and TX2 of the board.



As a result, the connection between these two should be like this.

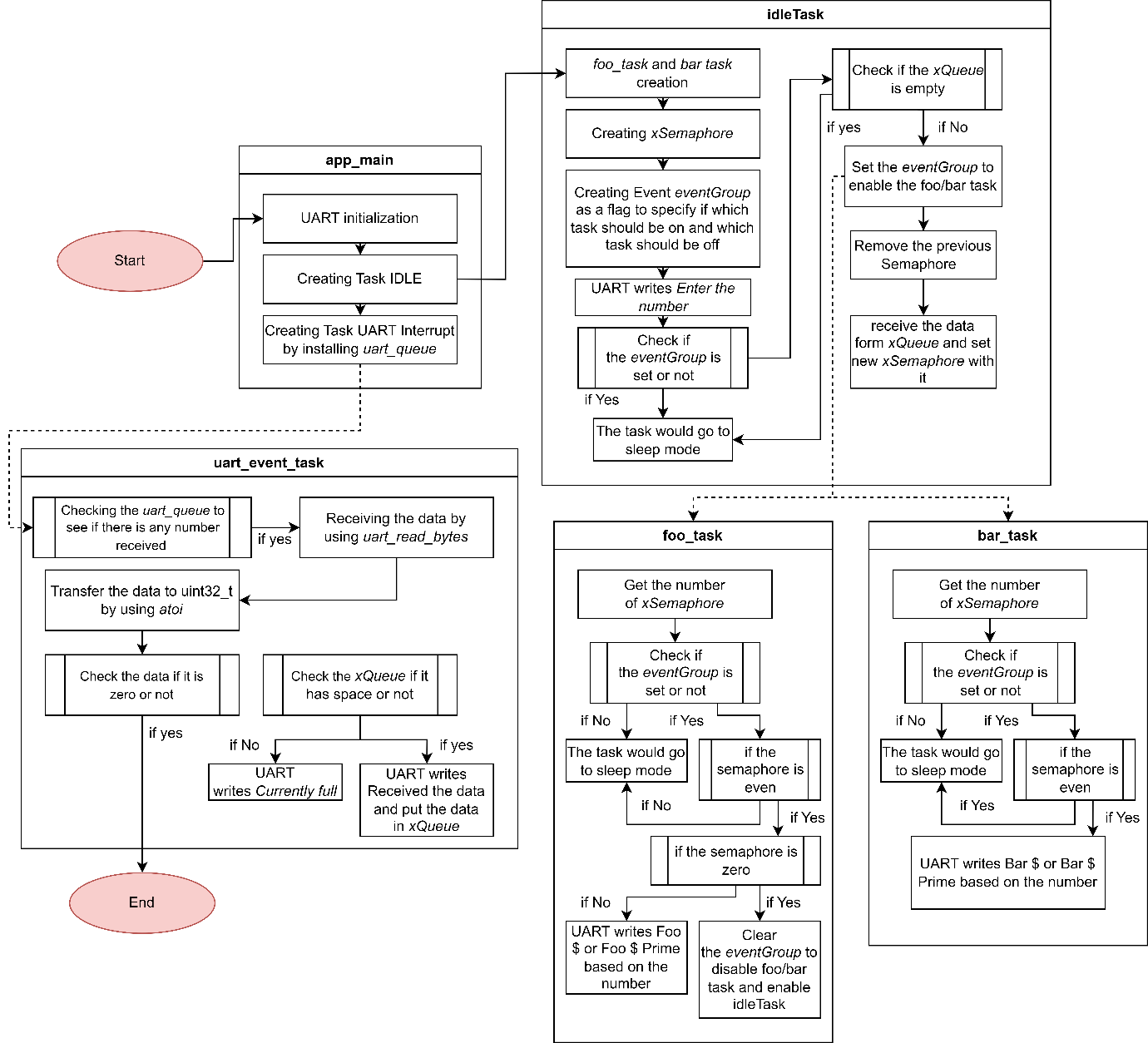


**Software Setup**

To build the code and flash the board, I have used VScode and Espressif IDF extension. To change the language programming, it is required to change the main file from .c to .cpp and change the both makefiles.

**Algorithm Setup**

The flowchart of the code is as follows

In this project, different types of FreeRTOS modules are used. In what follows, I have described how different RTOS are involved in this project.

1. Queue: we have two queues in this project; first, to receive the data from the Serial, *uart\_queue* is utilized. This queue is responsible for connecting all the received inputs from Serial. Next, because we need to have a storage of input numbers, *xQueue* is created. xQueue has 8 spaces to store the inputs based on the requirement of the project.
2. Semaphore: To count down the input number, *xSemaphore* is created. This can be easily checked by tasks which is odd or even.
3. Event: It is required to enable and disable the task by using flag. In the situation that foo and bar tasks are running, the idle tasks should be disabled and in the situation that there is no input number, foo and bar tasks should be disabled and only idle task should be running. That is why in my implementation, I have used Event to specify which task should be off and on.