Complete the “yellow” tabs and delate the phrases in italics.  
You can duplicate the table “Project”, if more than one project are due for the homework.

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A1* | | |
| Homework number: | HW10 | | |
| Due date: | 11/12 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| 1 *Giorgio Donato Carlo* |  |  | *x* |
| 2 *Lenzi Francesco* |  |  | *x* |
| 3 *Lodari Gianmarco* |  |  | *x* |
| 4 *Lanzini Alessio* |  |  | *x* |
| 5 *Chiapparo Lenn* |  |  | *x* |
| Notes:  *Complete in necessary* | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Keyboard 1a | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| To use the keyboard, we followed the steps proposed in the slides.  First of all we set the right input (PC2, PC3, PC12, PC13) and output (PC8, PC9, PC10, PC11) pins from the .ioc schematic.  For what matters the variables, we created 2 arrays containing the PIN addresses and 2 matrixes to save the previous and current keyboard statuses. In the main we created a matrix containing all the “key” values associated to every button of the keyboard.  In the while(1) cycle we activate one column and after 10ms (trough HAL\_DELAY) check all the rows by saving their status in one matrix, comparing the current value with the previous one and, if the button had a 0 to 1 transition we send its assigned “key” value through the UART. At the end of the cycle we update the vector containing the previous values of the keyboard status. After this we change column and repeat. | | | |
| Professor comments: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Keyboard 1b | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| We set up the Pins exactly like in the previous project. Additionally, we set TIM2 to trigger every 4 ms.  In HAL\_TIM\_PeriodElapsedCallback we read the pin state of the 4 buttons connected to one column. For each of them we check if its debounce time, which we set at 16 ms (the time to do a complete scan of the columns) has expired. To keep track of the debounce time of every button, we use a 4x4 uint16\_t matrix. If the time has expired it means that we can “trust” the new values, so we read and store it in “keypress”, a 4x4 matrix which contains 1 at the [row][col] position if the button [row][col] is pressed, otherwise 0.  We proceed by comparing the new button state with the old one (stored in an apposite matrix), if the button has switched from 0 to 1, it means that it has been pressed, so we proceed to transmit his character via USART.  After the transmission, the debounce time of this specific button is set to the default one(16 ms).  If the button debounce time was different than 0, we skip the read and transmission operations, and simply decrease it by one.  At the end, the column pin is put in RESET state, the column index is updated, and the next column pin is put in SET state. | | | |
| Professor comments: | | | |