|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B7* | | |
| Homework number: | *HW09* | | |
| Due date: | 29/11/2022 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Giacomo Massa |  |  | *x* |
| Simone Giampà |  |  | *x* |
| Luca Francesco Raduzzi |  |  | *x* |
| Johanna Micelli |  |  | *x* |
| Claudio Galimberti |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | HW\_09\_Encoder | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| SETUP: We enabled PC6 and PC7 in TIM3\_CH1 and TIM3\_CH2 since they’re the pins to where the encoder is connected. Then we configured timer 3 in encoder mode, which has the goal of basically counting the signals coming from the encoder.  We then configured TIM2 to get a timer of 1 second.  CODE:  The timer callback expires every second and is used to compute the rpm rotations of the encoder.  Then we defined 3 variables that contain respectively the last value computed by the encoder, the value computed in the previous iteration, and the difference between them.  We calculate a new value from the encoder and the new correspondent delta value, as the difference between the previous value and the new one. At each iteration, we check whether an underflow (or an overflow) on the delta value variable has occurred. These conditions are necessary for counting the rotations properly. As the last thing, we convert the delta value computed in rpm and send it to the terminal emulator via the usual UART DMA function. | | | |
| Question about the future keyboard project:  To read the keyboard matrix, we provide values for the columns and read values from the rows depending on the state of the button of each row.  Also, here we have NPN transistors, which are activated by pulling them to *vdd*. When we enable a column of buttons, by wiring them to *gnd*, if we press a button in that column, the corresponding row is pulled down from *vdd* to *gnd*. We read this transition as the press of a button.  We need to perform multiplexing by enabling one column at a time and looking for which row completes a transition from 1 (since it’s linked to *vdd* through a pullup resistor) to 0 (a situation in which we link it to *gnd* since the button is already pressed). From the code perspective, we would perform a cycle in which at each iteration we activate one single column and read all 4 rows, using respectively the GPIO functions HAL\_GPIO\_WritePin and HAL\_GPIO\_Reaadpin(). This way, we can scan the entire matrix of buttons. | | | |
| Professor comments: | | | |