CS443/543 - Embedded Systems

Assignment #4
Fall 2025

The purpose of this assignment is to implement a keypad scanner, using the keypad that was provided for the class.

This keypad is very simple - it consists of 16 buttons, organized in rows and columns. Each key is located at the intersection of a row and column. When a key is pressed, the particular row and column are connected together. Most keyboards also have the same functionality at the lowest level; however, they usually employ some circuitry that senses the intersection, decodes the particular key being pressed, and then sends out a code that identifies that key. This assignment will implement that functionality.

Assume that pins connected to rows will be outputs, and pins connected to columns will be inputs. The idea behind a scanner is as follows: first, one of the rows is asserted, then each column is read, to see if a connection exists. If it does, that indicates a key being pressed; if not, then the scan continues. This process is repeated, activating successive rows and reading each column, until either a key press is identified, or the scan is completed without detecting any keys being pressed.

The term ”asserted” as used above indicates that a row has been selected, without indicating the polarity of that selection. If the input (column) pins have the internal pullup enabled, then they will be read as a ’1’ if unconnected. Thus, in order for an input pin to detect an assertion, the corresponding output pin should be asserted as a ’0.’

So the algorithm should do as follows:

1. Assert (set to ’0’) the pin connected to row 1.
2. Read each column individually.
    - If a column pin reads a ’0’, then the key at that row/column is the scanned key.
    - If the column pin is a ’1’, then continue the scan.
3. - If no key presses are detected, then assert the pin connected to the next row and repeat.

It is recommended that the algorithm be enclosed in a function which returns the value (ASCII or numeric) of the key that was pressed.

The keys on the keypad are pretty loose, and are probably very susceptible to bouncing. The algorithm above should also include de-bouncing.