CP320 EXPLORATION DOCUMENTATION Tinotenda Nyakurerwa and Josh Grift

Overview:

This document contains information on how the we were able to use a 3x4 matrix keypad(shown below) for the first time.

A close up of a building

Description automatically generatedA close up of a device

Description automatically generated

Packages Needed:

* RPi.GPIO
* Time

Circuit overview:

The circuit used was composed of a Raspberry Pi and 3x4 keypad matrix. The keypad was connected directly to the Pi and no other materials were used. The packages used were the RPi.GPIO package to allow control of the GPIO on the Raspberry Pi. The time package was also used, this package was needed to allow the program time to process the value of the key pressed. The software figures out which key has been pressed by first scanning through all the keypad rows for an input then once the row value is found the row value is stored, the software then proceeds to loop through all the columns to find the column value of the pressed key. A combination of the column and row value were then used to provide the value of the key pressed.

Challenges:

Initially we attempted to write the code on our own after having read up on how the matrix works and how to set it up. However, because we had not used the pull-up and pull-down resistors, the value of the keys just kept on randomly changing when only one key had been pressed. We then searched the internet and found code that did not have any other components to it except for the keypad matrix. This code was found at the following link <https://crumpspot.blogspot.com/2013/05/using-3x4-matrix-keypad-with-raspberry.html?m=1> .

We modified the code that we found from taking in just one key value at a time, to being able to take an infinite number of key values and stopping only if the user presses the “\*” or “#” key. This then presented us with a new problem of having the keypad only taking in one key value and printing that infinitely despite other keys being pressed, this issue was fixed by importing the time package and using the sleep() function after the program gets the key press, this then provided the program with ample time to process each key value. We also modified the original code by implementing a try except block, this is to allow the user to escape using the normal command line escape of Ctrl-C in the event that they are unable to stop the program or something unexpected happens and the program does not run as intended.

Additional links used:

* This link helped us with the initial understanding of how keypad worked: <https://tutorials-raspberrypi.com/connecz-raspberry-pi-kecpad-code-lock/>
* This link led us to the forum that led us to finding the code that was used: <https://www.raspberrypi.org/forums/viewtopic.php?t=34235>