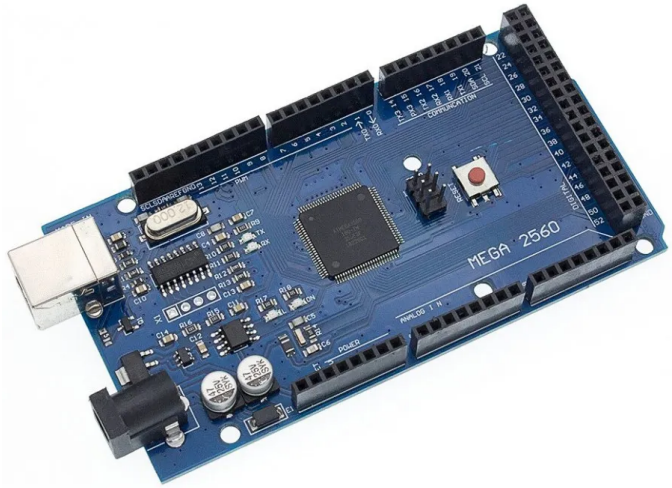
Mini Games Documentation

Motivation

While conducting research for this assignment, I came across the concept of implementing popular mini-games on an Arduino Board. Therefore, upon much thinking, I decided to create 2 which were dearest to me during my childhood. Having played them on old Nokia phones, Pac-Man and Hangman represented the height of technology at that time to me. Having said that, I do have to recognize that these come in a simpler version due to the limitations imposed by the available components.

Necessary Components

Arduino Mega 2560



Specifications

LCD Keypad Shield 1602

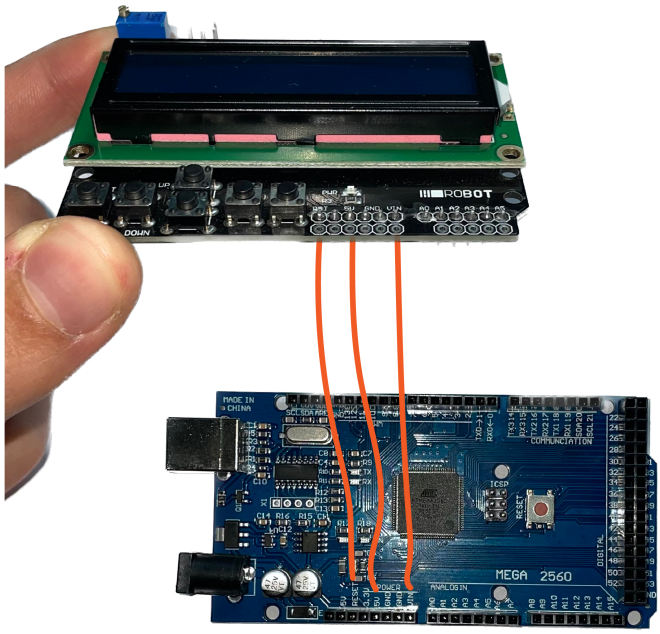
Specifications

USB 2.0 Cable Type A/B

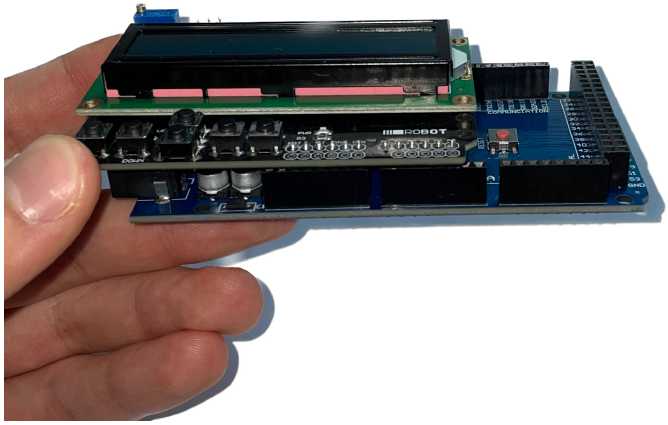
Specifications

Assembly

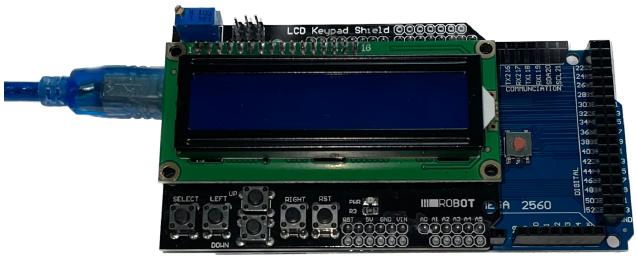
Step 1 - Attach the LCD to the Arduino Board making sure the 5v overlaps the designated 5v hole in the board.



Step 2 - The 2 pieces are attached



Step 3 - Attach the USB cable to the board and connect it to the computer



Step 4 - Upload the code from the Arduino IDE to the board (make sure the correct board and port are selected in the “Tools” section)

Games Development and Description

To achieve all of these games, the use of the Liquid Crystal Library, a widely-used Arduino library, was essential for controlling the display of everything on the screen. The LiquidCrystal library enables an Arduino/Genuino board to control LCD displays based on the Hitachi HD44780 or compatible chipset, supporting both 4-bit and 8-bit modes of operation. [1]

Pac-Man

Description

The project aims to implement a simplified version of the classic Pac-Man game, optimized for small display sizes. The game allows the user to control Pac-Man’s moves using unidirectional buttons: up, down, left, and right. Such input enables the character to move through the maze and collect food while avoiding the ghosts. After

eating all the food points on the map, the player levels ups while the game’s difficulty increases. This means that the ghost becomes faster and faster as the user advances through the game. Nonetheless, when being caught by the. ghost, the game ends by displaying the total amount of points gained.

Implementation

Several constants have been defined in order to specify important parameters such as Pac-Man’s and the ghost’s speed, screen dimensions, and control buttons. Moreover, keeping track of PacMan’s, the Ghost’s, and the food points’ positions, as well as the game’s level and score, was done through the use of designated variables.

The main functions of the program are as follows:

1. “Move” is responsible for handling Pac-Man's movement on the screen by updating its position based on user input.
2. “Pursuit” manages the movement of the ghosts, determining their behavior and positioning in relation to Pac-Man's movement. In order to do this, a simple “AI” algorithm was used, which assures that the ghost changes its vertical position when the PacMan does.
3. “GameOver” is called when Pac-Man is caught by a ghost, indicating the end of the game.
4. “Win” is by successfully eating all the food points.
5. “InitLevel” initializes the current game level by setting up the maze, the layout, and the food points on the screen.

Hangman

Description

The project aims to implement the classic Hangman game, optimized for small display sizes. The way in which the game actually works is that the program contains a predefined list of words from which one is randomly selected. The player navigates through the alphabet, using the “left” and “right” buttons, and selects the desired

letter by pressing the “select” button. When a letter is selected, the program displays an asterisk (*) in place of the letter, and the player is not allowed to select that letter again. When correctly guessing a letter, it is displayed in its designated position in the word below (if there are 2 or more appearances of that letter, all will be completed). Nonetheless, If the player correctly guesses the word, they win and the game board is reset. If too many mistakes are made, aka the entire stick figure is drawn, the game ends, and the board is reset by holding the “select” button down for 2 seconds.

It also includes variables for the guessed letter, the last guessed letter, a string for previously guessed letters, a string for the secret word, a counter for guessed letters, variables to check if a letter has been guessed or guessed previously, variables for displaying the asterisk, button state, number of remaining attempts, number of correctly guessed letters, number of selections, last pressed button, and last debounce time. The code also includes custom characters for displaying the guillotine graphics.

The main functions of the program are as follows:

1. "Setup" initializes the LCD, and sets the custom characters for displaying the guillotine graphics
2. "Loop" checks if a button has been pressed and makes the corresponding decision. If the select button is pressed, it checks if the correctly guessed letters have reached the required number to guess the word or if the number of attempts has reached its maximum, and makes the appropriate decision (win or loss).

References

[1] https://www.arduino.cc/reference/en/libraries/liquidcrystal/

[2] **https://docs.arduino.cc/learn/electronics/lcd-displays**

[3] **https://howtomechatronics.com/tutorials/arduino/lcd-tutorial/**

[4] **https://www.instructables.com/PacMan-and-custom-characters-on-Arduino-with-a-16x/**