HexMatch Design Document

Date: 12/08/2023

Author: Garrett Engelder

Table of Contents

Overview

HexMatch is a game implemented on a Raspberry Pi 4, utilizing various hardware components, including TM1638 and TM1637 modules, 1-bit 7-segment LED display, LEDs, and a momentary switch. The game challenges the player to match randomly generated hexadecimal symbols/characters displayed on the TM1638 module with a target symbol/character on the 1-bit 7-segment LED display. The player must successfully complete 10 matches in a row to win the game.

Hardware Setup

TM1638 Module:

- **VCC**: Physical/Board pin 1 (3v3)
- **GND**: Physical/Board pin 9 (GND)
- **STB**: Physical/Board pin 15 (GPIO/BCM pin 22)
- **CLK**: Physical/Board pin 40 (GPIO/BCM pin 21)
- **DIO**: Physical/Board pin 11 (GPIO/BCM pin 17)

TM1637 Module:

- **CLK**: Physical/Board pin 5 (Wiring Pi pin 9)
- **DIO**: Physical/Board pin 29 (Wiring Pi pin 21)
- **VCC**: Physical/Board pin 17 (3v3)
- **GND**: Physical/Board pin 39

1-bit LED Display:

- **Segment A**: Physical/Board pin 8 (Wiring Pi pin 15)
- **Segment B**: Physical/Board pin 10 (Wiring Pi pin 16)
- **Segment C**: Physical/Board pin 12 (Wiring Pi pin 1)
- **Segment D**: Physical/Board pin 16 (Wiring Pi pin 4)
- **Segment E**: Physical/Board pin 18 (Wiring Pi pin 5)
- **Segment F**: Physical/Board pin 22 (Wiring Pi pin 6)
- **Segment G**: Physical/Board pin 24 (Wiring Pi pin 10)
- +: Physical/Board pin 20 (GND)

Hardware Setup Cont.

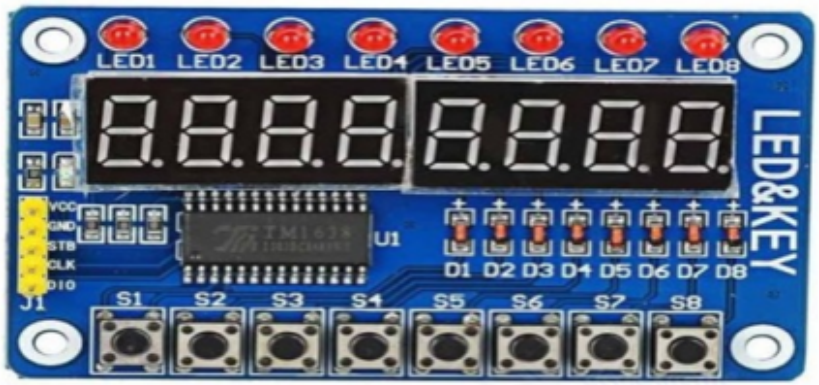
Green, Red, Yellow LED:

- **GND**: Physical/Board pin 34
- **Green LED:** Physical/Board pin 19 (Wiring Pi pin 12)
- **Red LED:** Physical/Board pin 21 (Wiring Pi pin 13)
- **Yellow LED:** Physical/Board pin 23 (Wiring Pi pin 14)

Momentary Switch:

- **VCC**: Physical/Board pin 17 (3v3)
- **GND**: Physical/Board pin 14
- **Signal**: Physical/Board pin 31 (Wiring Pi pin 22)

HexMatch Connectivity Diagram



Physical Pin 1
Physical Pin 9
Physical Pin 15
Physical Pin 40
Physical Pin 11

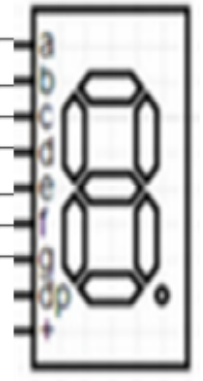
CLK — Physical Pin 5
DIO — Physical Pin 29
VCC — Physical Pin 17
GND — Physical Pin 39

Raspberry Pi 4 Model B

	3V3	1 2	5V		
I2C SDA	GPIO2	3 4	5V		
I2C SCL	GPIO3	5 6	GND		
	GPIO4	7 8	GPIO14	UART TX	
	GND	9 10	GPIO15	UART RX	
	GPIO17	11 12	GPIO18	PCM CLK	PWM0
	GPIO27	13 14	GND		
	GPIO22	15 16	GPIO23		
	3V3	17 18	GPIO24		
SPI MOSI	GPIO10	19 20	GND		
SPI MISO	GPIO9	21 22	GPIO25		
SPI SCLK	GPIO11	23 24	GPIO8	SPI CE0	
	GND	25 26	GPIO7	SPI CE1	
I2C ID EEPROM	GPIO0	27 28	GPIO1	I2C ID EEPROM	
	GPIO5	29 30	GND		
	GPIO6	31 32	GPIO12	PWM0	
	PWM1	GPIO13	33 34	GND	
PWM1	PCM FS	GPIO19	35 36	GPIO16	
		GPIO26	37 38	GPIO20	PCM DIN
		GND	39 40	GPIO21	PCM DOUT

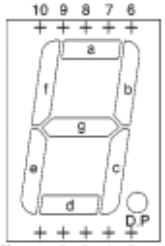
All seven resistors = 220 ohms 1/8 watt

Physical Pin 8
Physical Pin 10
Physical Pin 12
Physical Pin 16
Physical Pin 18
Physical Pin 22
Physical Pin 24



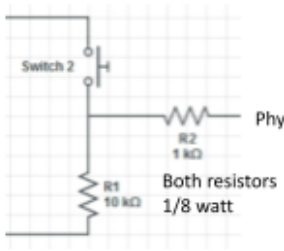
Physical Pin 20 to display pins 3 & 8 (common cathode pins)

LA-6780 Common Cathode 7 Segment Display



Pin No.	Function
1	Segment "e"
2	Segment "d"
3	Common
4	Segment "c"
5	D.P
6	Segment "b"
7	Segment "a"
8	Common
9	Segment "f"
10	Segment "g"

Physical Pin 17



Physical Pin 14

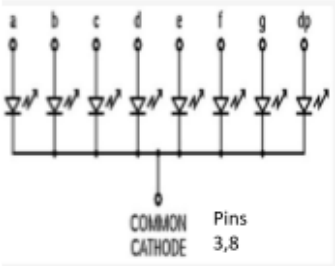
Both resistors 1/8 watt

Pi Pin 34

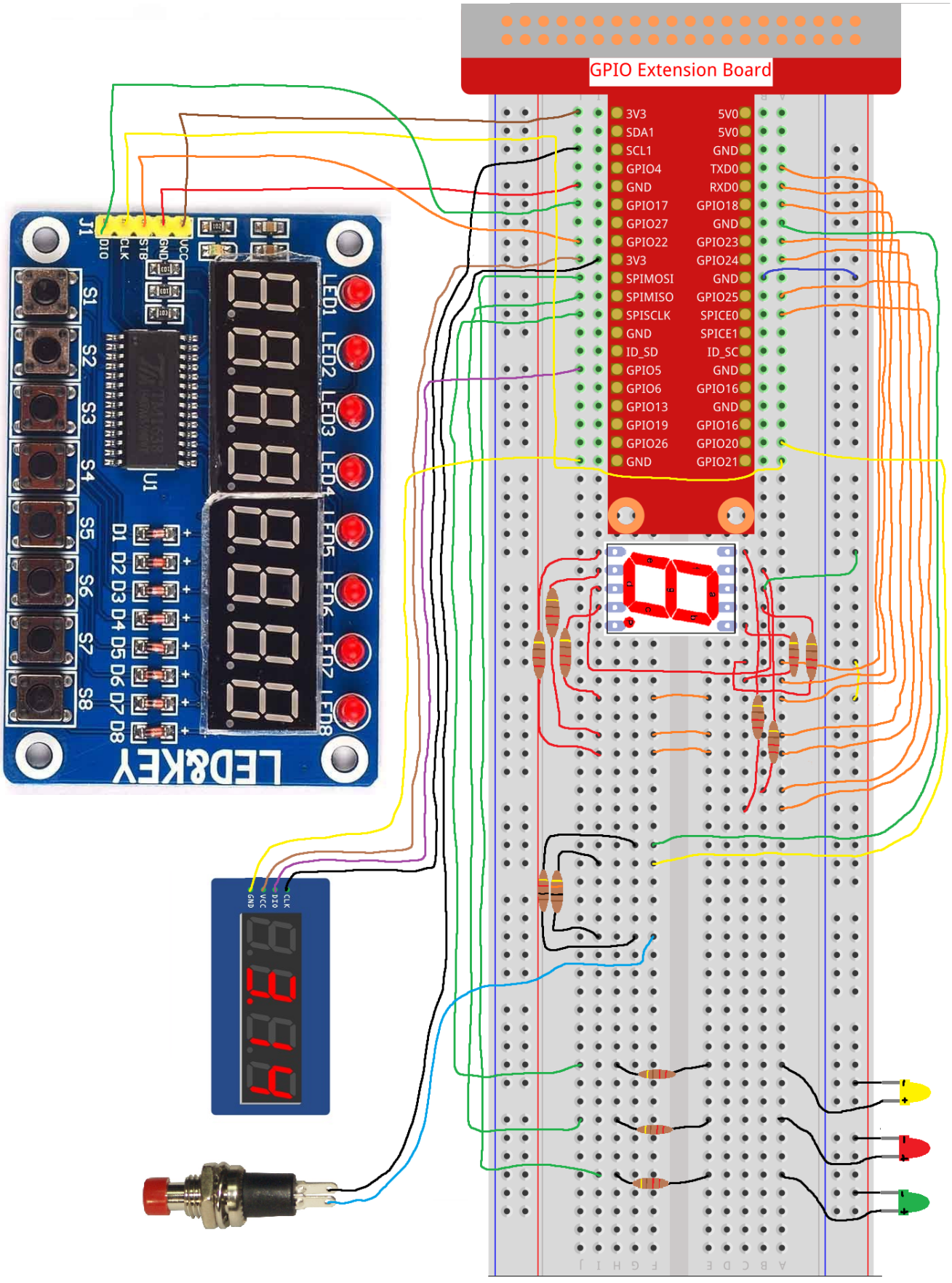
Physical Pin 19 — Green LED = Success
Physical Pin 31
Physical Pin 21 — Red LED = Failure
Physical Pin 23 — Yellow LED = Playing

All three resistors = 220 ohms 1/8 watt

Switch 2 pressed to reset game. 1kohm current limiting resistor used in case GPIO accidentally set to output.



Important: The Raspberry Pi GPIOs support voltages only up to 3.3V. If you attach a higher voltage, you may permanently damage your Raspberry Pi.



HexMatch Wiring Diagram

Required Software Libraries

- **wiringPi library:**
  - Installation: WiringPi is PRE-INSTALLED with standard Raspbian systems.
- **bcm2835 library & header file:**
  - Installation: This library consists of a single non-shared library and header file installed by make install

```
1. tar zxvf bcm2835-1.73.tar.gz
2. cd bcm2835-1.73
3. ./configure
4. make
5. sudo make check
6. sudo make install
```

  - https://www.airspayce.com/mikem/bcm2835/
- **tm1638 header file:**
  - Installation: You can grab the code from github here: https://github.com/mjoldfield/pi-tm1638 OR from the provided tm1638 program folder.
  - Open a terminal on your Raspberry Pi. (needs autotools):

```
1. git clone
   https://github.com/mjoldfield/pi-tm1638.git
2. autoreconf -vfi
3. ./configure
4. make
5. sudo make install
```

- Open a terminal on your Raspberry Pi. (no autotools needed):

```
1. wget
   https://github.com/downloads/mjoldfield/pi-tm1638
   /pi-tm1638-1.0.tar.gz
2. tar xzvf pi-tm1638-1.0.tar.gz
3. cd pi-tm1638-1.0
4. ./configure
5. make
6. sudo make install
```

- **tm1637 header file:**
  - Installation: You can grab the code from: https://framagit.org/harlock/tm1637-c-library-for-raspberry-pi OR from the provided tm1637 program folder.
  - Open a terminal on your Raspberry Pi.:

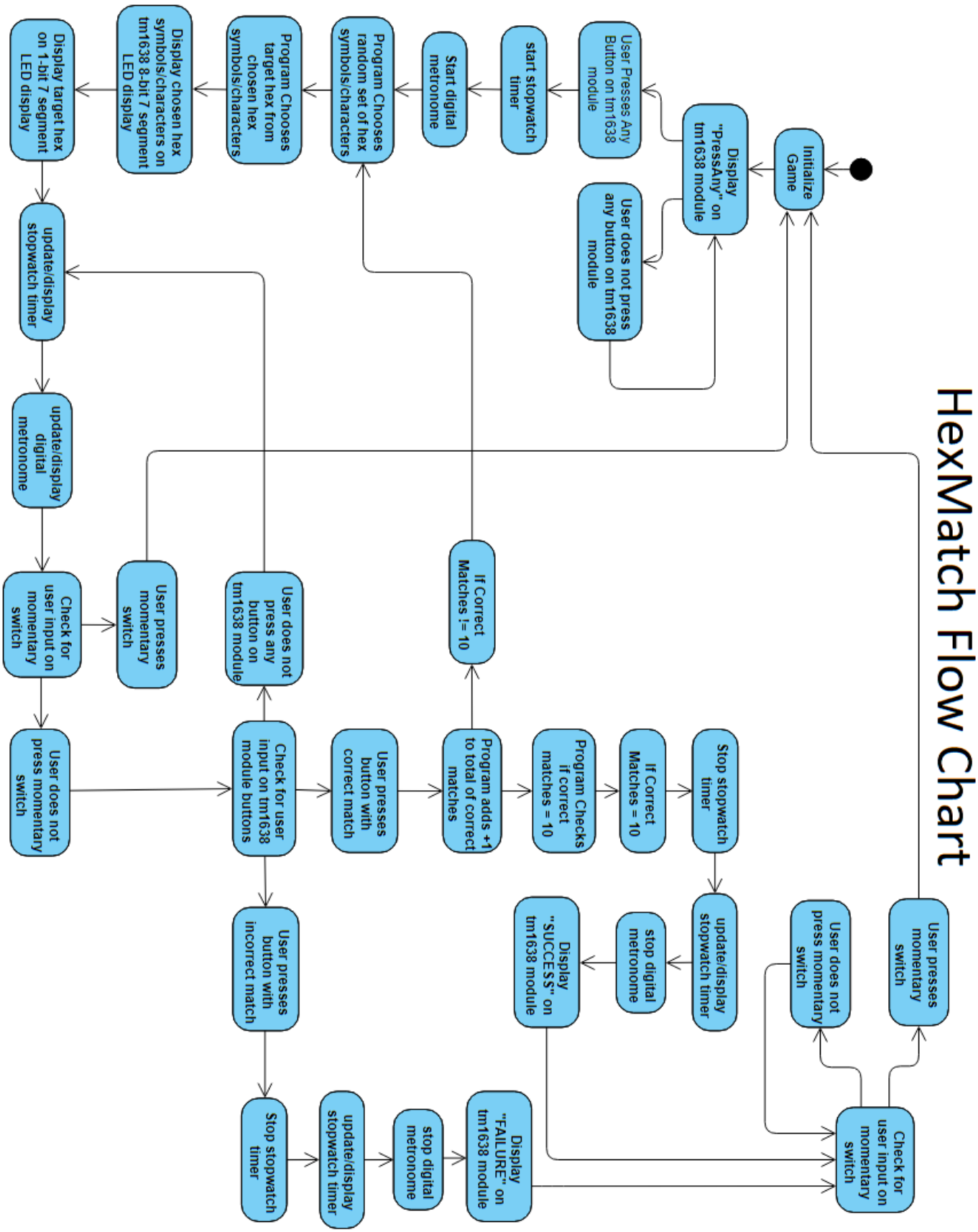
```
1. sudo su
2. mv tm1637.h /usr/include/
3. ls /usr/include
4. exit
```

Game Flow

1. **Initialization:**
    - Initialize hardware and software components.
    - Display "PressAny" on the TM1638 module.
2. **Game Start:**
    - Player presses any button on the TM1638 module.
    - Display 8 randomly chosen unique hexadecimal symbols/characters on the TM1638 module.
3. **Game Setup:**
    - Choose a target hexadecimal symbol/character randomly from the 8 randomly chosen unique hexadecimal symbols/characters on the TM1638 module.
    - Start a stopwatch timer on the TM1637 module.
    - Activate a digital metronome on the TM1638 module.
4. **Gameplay:**
    - Player must match the target symbol/character on the 1-bit LED display by pressing the corresponding button on the TM1638 module.
    - Repeat the matching process 10 times in a row without failure.

Game Flow Cont.

5. **Game Completion:**
    - **If the player successfully matches the symbols 10 times:**
        - Turn off the digital metronome and 1-bit LED display.
        - Display "SUCCESS" on the TM1638 module.
        - Light up the green LED.
        - Stop the stopwatch timer and display the elapsed time on the TM1637 module.
    - **If the player fails to match a symbol:**
        - Turn off the digital metronome and 1-bit LED display.
        - Display "FAILURE" on the TM1638 module.
        - Light up the red LED.
        - Stop the stopwatch timer and display the elapsed time on the TM1637 module.
6. **Restart Option:**
    - Player can press the momentary switch at any time (except during "PressAny") to restart the program.


HexMatch Flow Chart
Initialize Game
Display "PressAny" on tm1638 module
User Presses Any Button on tm1638 module
User does not press any button on tm1638 module
start stopwatch timer
Start digital metronome
Program Chooses random set of hex symbols/characters
Program Chooses target hex from chosen hex symbols/characters
Display chosen hex symbols/characters on tm1638 8-bit 7 segment LED display
Display target hex on 1-bit 7 segment LED display
update/display stopwatch timer
update/display digital metronome
Check for user input on momentary switch
User presses momentary switch
User does not press any button on tm1638 module
Check for user input on tm1638 module buttons
User presses button with correct match
User does not press momentary switch
If Correct Matches != 10
Program adds +1 to total of correct matches
Program Checks if correct matches = 10
If Correct Matches = 10
Stop stopwatch timer
update/display stopwatch timer
stop digital metronome
Display "SUCCESS" on tm1638 module
User presses momentary switch
User does not press momentary switch
Check for user input on momentary switch
Display "FAILURE" on tm1638 module
stop digital metronome
update/display stopwatch timer
Stop stopwatch timer
User presses button with incorrect match

Program Structure

Main Program:

- Initializes hardware components.
- Waits for the player to press any button to start the game.
- Manages the main game loop.

Game Functions:

- setup(): Configures initial setup, sets up pins for LEDs and buttons.
- chooseRandomChars(): Generates an array of 8 unique random hexadecimal symbols/characters.
- displayChosenHex(): Displays the chosen hexadecimal symbols/characters on the TM1638 module.
- update1BitLED(): Updates the 1-bit LED display based on the target hexadecimal symbol/character.
- displayTimer(): Displays the elapsed time on the TM1637 module.
- checkButtonPress(): Checks for button presses and handles the game logic.
- gameCompletionCheck(): Checks if the player successfully completed the game.
- gameCompletionFailure(): Handles game over scenario.
- restartProgram(): Restarts the program.

Development Challenges and Current Issues

Initial Library Integration:

One of the initial challenges I faced was determining the necessary libraries to integrate the tm1638 and tm1637 modules with the Raspberry Pi. This process involved trial and error, exploring various libraries, and understanding their dependencies. Fortunately, I overcame this hurdle by identifying and successfully integrating the bcm2835 and wiringPi libraries for the tm1638 and tm1637 modules, respectively. The code now reflects a cohesive integration of these libraries.

Library Referencing Discrepancy:

Another challenge emerged when dealing with the discrepancy in library referencing for the tm1637 and tm1638 modules. The tm1637 utilized the wiringPi library, consistent with other components like the momentary switch, the 1-bit 7-segment LED display, and the game status LEDs. However, the tm1638 module required the bcm2835 library, leading to confusion in pin referencing compared to other parts of the code.

Development Challenges and Current Issues Cont.

Threading Implementation:

A current challenge revolves around implementing threading into the code. Threading is essential for achieving concurrent execution of the stopwatch timer, digital metronome, and button press checking loops. The absence of threading introduces delays in these processes, impeding the addition of new and altered features such as game music playback and the display of milliseconds on the stopwatch timer. Overcoming this challenge will not only enhance the game's responsiveness but also enable the incorporation of additional dynamic features.

Future Enhancements:

1. **Concurrent Execution:** Threading is essential for simultaneous execution of critical game components, such as the stopwatch timer, metronome, and button press checking loops.

2. **Dynamic Features:** With threading in place, the door opens for implementing dynamic features like game music during play and displaying milliseconds on the stopwatch timer.

3. **Reduced Delays:** Threading will help minimize delays caused by sequential execution, providing a smoother and more responsive gaming experience.

Addressing the threading issue will not only resolve the current challenges but also pave the way for further enhancements, elevating the overall HexMatch game experience.

Conclusion

The HexMatch game provides an engaging challenge for players to test their memory and reaction time. The combination of hardware components and a well-defined game flow creates an immersive and enjoyable gaming experience on the Raspberry Pi 4.