DMP LAB
MINI MARIO PROJECT

Pașc Andreea-Mădălina

3rd Year of Computer Science

Group 30431

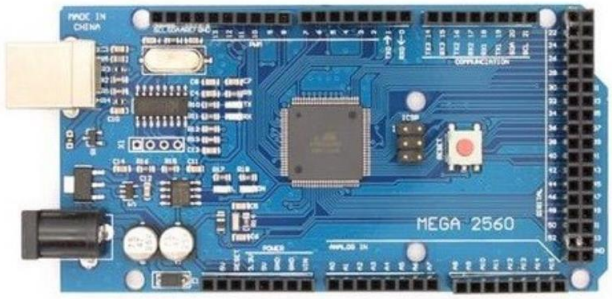
1. Project Description

The main objective for this project is to simulate a mini Mario Game. In the game, Mario has to jump over a pipe using the two buttons. If he is unable to jump, then the game is over and the overall score is the total play time in seconds.

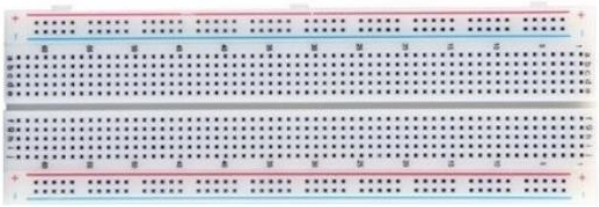
2. Components

For this project, the main components are:

- The Mega 2560 Board



- A breadboard



- A 2004 LCD with I2C

- 2 buttons

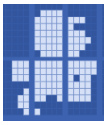
- Wires

3. Implementation

The logic for this project's implementation was split into multiple parts. First, we try and draw the frames for the Mario character, then we try to make him move. After that we need the obstacle: a pipe and we need to make him jump and come back on the ground. The frames must be cleaned in between and after the jumps. If Mario hits the pipe, then the game is over and the score will we printed on the screen along with a message. The score will be the total time in seconds since the start of the game and the collision with the pipe.

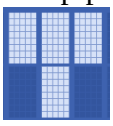
- The LCD
  - The LCD is connected using the I2C with the pins GND, VCC, SDA and SCL on the board
  - It works using the "LiquidCrystal_I2C.h" library and it is initialized with the lcd.init() function and lcd.backlight() function

- The Mario character:



  - The frames for the Mario character were taken from https://forum.arduino.cc/index.php?topic=203873.0 , only the first to frames were used in order to make him look like he is running.
  - In total there are 12 frames: 6 frames for when he is standing and 6 frames with his leg in the air. Alternating between these 2 frames gives the illusion of running
  - The frames were put on the LCD screen using the functions lcd.createChar(id, string), lcd.setCursor(column, row) and lcd.write(id)
  - Every frame has a different id and it is printed on the screen using that specific id

- The pipe:



  - The pipe is meant to be the obstacle for Mario
  - Mario is actually running in place and the pipe is moving from the end of the LCD towards him giving the impression that he is actually running for the pipe
  - The pipe is moving its place using a for instruction that initiates its position at the end of the screen until the beginning of the screen and so on
  - In order to clean after the pipe so that the screen behind it is clear, we made an empty character string and we introduce it after each iteration of the pipe

- The jumps
  - In order to control the jump we use the 2 separate buttons: one for jumping and one for coming back on the ground

- The buttons work using system interruption and are put on pin D2 and D3 on the Mega board
- The interrupts are done using the attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);
- The functions for the jumps are used in order to set the Boolean values jumped and clearMario true or false and to set the row for Mario
- If button 2 is pressed than Mario jumps, the jumped and clearMario Boolean are set to true so that Mario can be printed on rows 0 and 1 without any image residue behind him and the row is set to 0 not to mess up with the collision
- If button 3 is pressed than Mario comes back on the ground and the Boolean values are set to false and the row is set to 2
- This logic helps when Mario is in the air so that we can draw the pipe under him without problems or bugs from other empty characters that are being printed for cleaning the screen

```
// jumps
void jump(){
  jumped = true;
  clearMario = true;
  row = 0;
  }

void jumpEnd(){
  jumped = false;
  clearMario = false;
  row = 2;
  }
```

- Clearing the screen:
  - Clearing the screen after jumps is done by printing multiple empty characters “ “ on the screen instead of the frames

- The collision:
  - The collision is done by calculating the distance between the first frame of the pipe and Mario’s head frame and by verifying if the row that Mario is in is 2, and implicitly 3

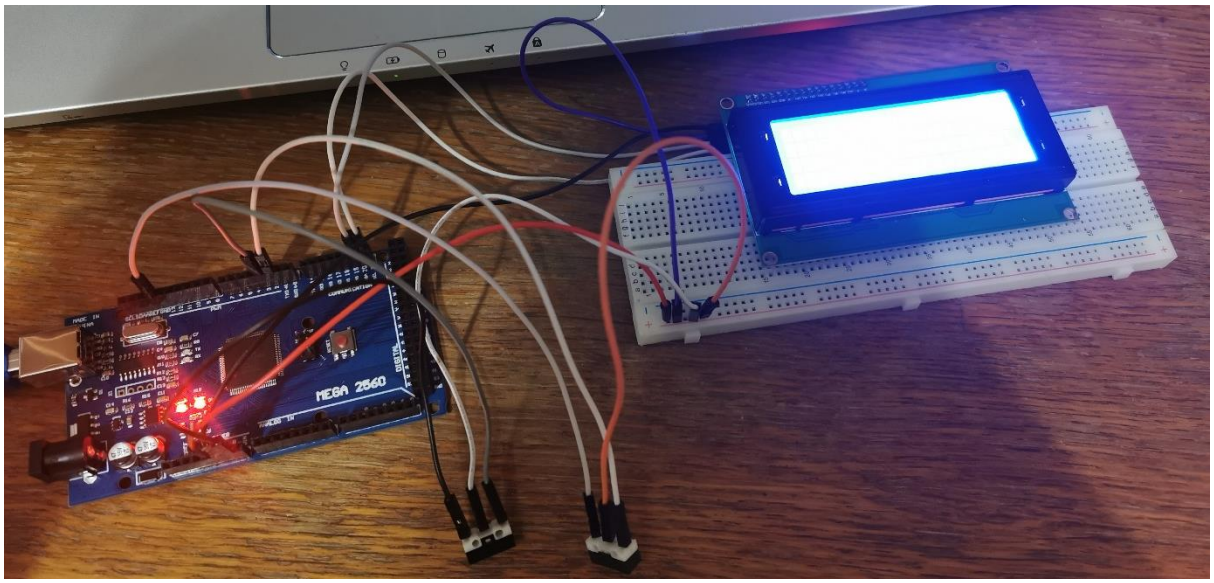
```
// collision
distance = pos - 1 - c;
if(row == 2 && distance == 0){
  endGame();
  }
```

- End of the game and measuring the score:
  - The end of the game happens when a collision happens
  - After a collision, the endGame() function is being called, the Boolean value play is set to false so the game cannot continue, the screen is cleared and a message with “Game over” and the score appears
  - The score is measured as the time in seconds between the start of the game and the moment of collision
  - In order to measure the time we use use the function millis() and we divide it by 1000

- The score is also shown during the game in the top right corner of the LCD

```
// game over
void endGame(){
  play = false;
  lcd.clear();
  lcd.setCursor(4, 1);
  lcd.print("Game Over");
  lcd.setCursor(3, 2);
  lcd.print("Your score: ");
  lcd.setCursor(16 ,2);
  lcd.print(score);
  }
```

- The setup:



4. Conclusion

In conclusion, this was both a fun and educational project to work on. It was a very helpful project that helped to understand better how to work using the Mega 2560 board and the Arduino functions in order to print and control characters on the screen. For further development, it would be nice if the pipes were more and if they appeared randomly on the ground and on the top 2 rows and also if besides time, Mario could collect random coins on the screen, have different lives and maybe some powers.

5. Bibliography

- https://forum.arduino.cc/index.php?topic=203873.0
- DMP labs and courses
- https://www.arduino.cc/en/Reference/LiquidCrystalCreateChar