Proiectare cu MicroProcesoare - Project
- 2048 with Gyroscope -

Donică Ion
Grupa: 30237

15/12/2025

FACULTATEA DE AUTOMATICA
SI CALCULATOARE

Contents

1 Introduction

The integration of technology into the education system has accelerated, presenting nearly limitless opportunities for fostering cognitive development in children of all ages.

Technology has the potential to enhance various skills, including language proficiency, fine motor skills, memory, and attention, making it an invaluable tool in educational settings.

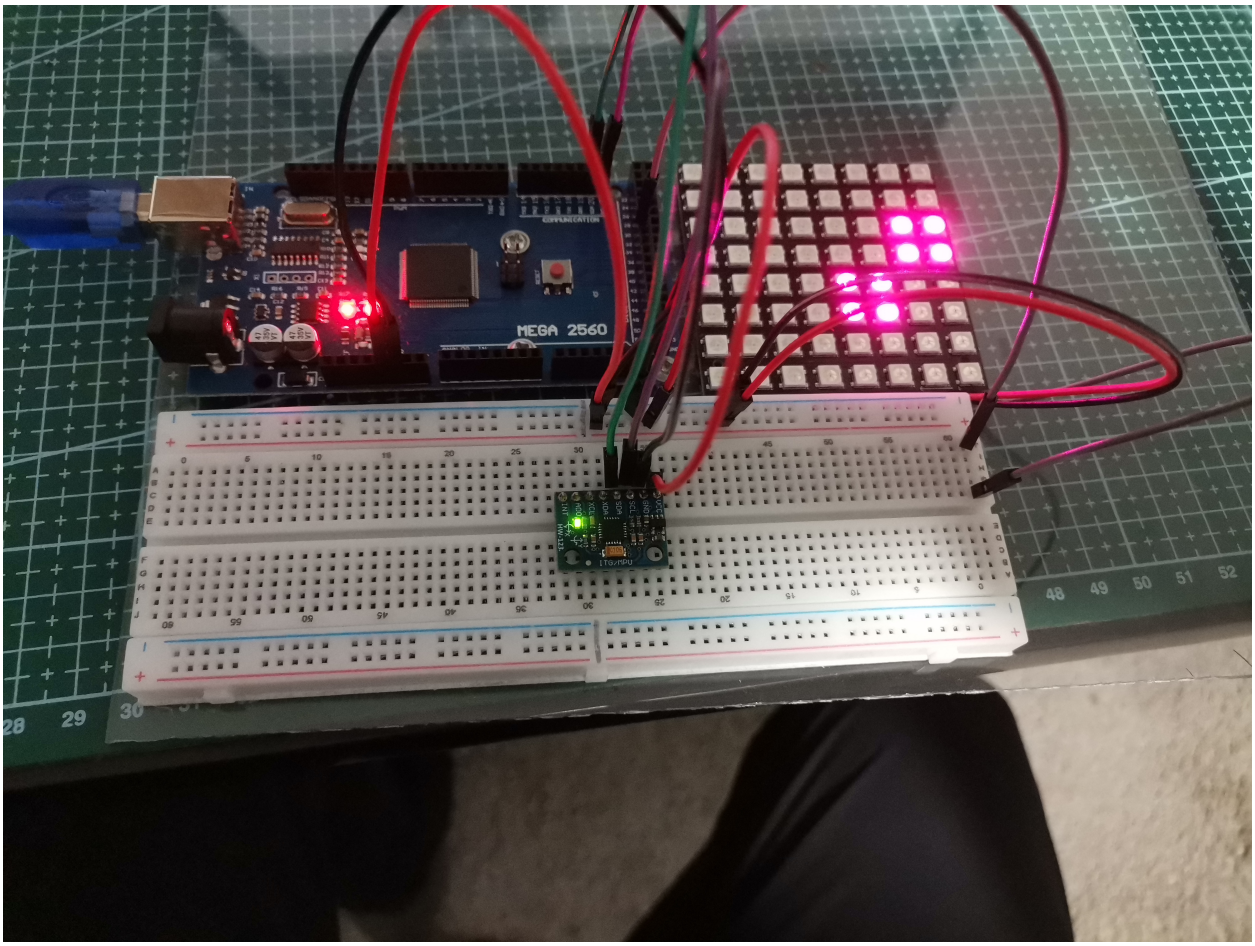
In light of this potential, this project proposes a solution aimed at the special education sector, specifically benefiting children with sensory and motor skill challenges, poor lateralization, and even older adults recovering from strokes.

The project employs a gyroscope sensor to detect inclination, calculate the state, and update the display, thereby illustrating key principles to children. Communication with the board is facilitated through the $\mathbf{I^2C}$ protocol.

To streamline the interaction between these components, predetermined libraries have been utilized.

To capture and maintain children's interest more effectively, an RGB palette has been chosen.

The primary goal is to enhance the therapy process in occupational therapy for children with special needs and to support their developmental progress.



2 Hardware

2.1 Components

2.1.1 Main Board

For the main board, an Arduino mega2560 was used. The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button [**arduino_2023_mega**].

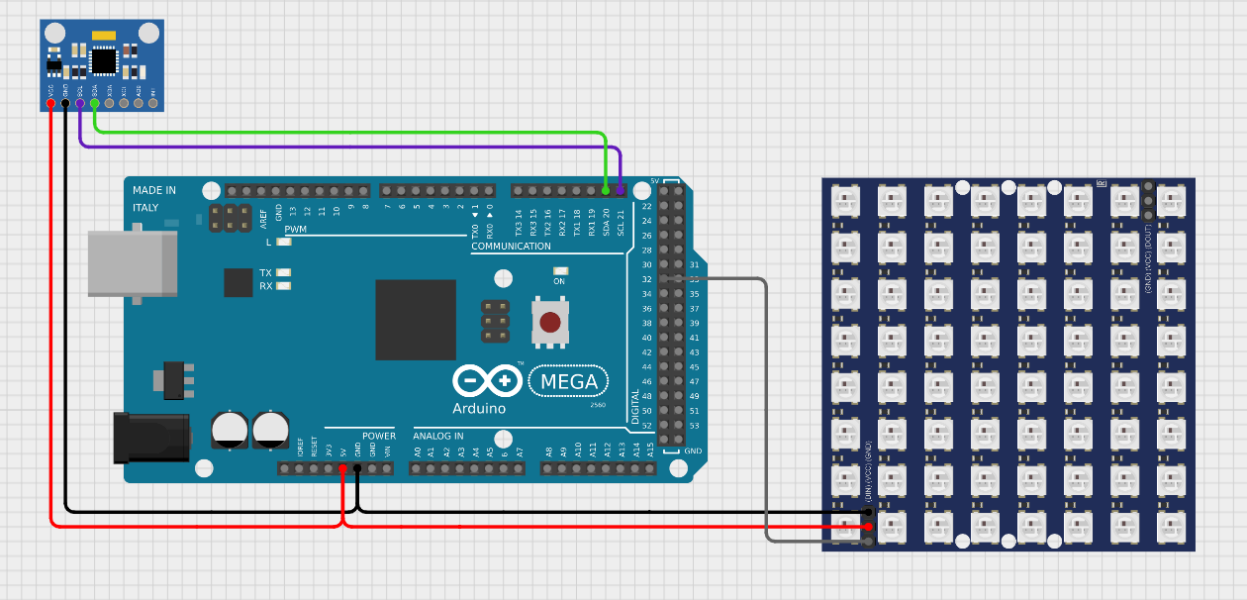
2.1.2 Input

For the input, an **MPU6050** was used. The **MPU6050** devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die, together with an onboard Digital Motion Processor™ (DMP™), which processes complex 6-axis MotionFusion algorithms. The device can access external magnetometers or other sensors through an auxiliary master **I²C** bus, allowing the devices to gather a full set of sensor data without intervention from the system processor. [https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/]

2.1.3 Display

For the display, an **WS2812B** NeoPixel 8x8 Led Strip was used. Even if it is called a LED Strip, it is actually a matrix, but in the code is is still refered as an array.

2.2 Diagram



2.3 Connections

This symbol "**$**" will be used to indicate what color the pin has on the diagram so it's easier to follow along.

The common ports that both components use are **$VCC** and **$GND** . These connect to the dedicated **$VCC** and **$GND** ports on the board.

Out of the 8 ports the *input* has, only 4 are of intereset to us: **$VCC** , **$GND** , **$SCL** and **$SDA** .

$SCL (**clock**) is connected to digital port **21 (SCL)**
$SDA (**data**) is connected to digital port **20 (SDA)**
These are the ports of the **MPU6050** which communicate with the main board using the **I^2C** protocol.

The *display* has 3 ports: **$VCC** , **$GND** and **$DIN** .
$DIN is connected to digital port **32**, but it is configurable.
This is the port that controls the display, and where the data is sent.

3 Software

3.1 Code implementation

3.1.1 2048.h

```
#ifndef _2048_H
#define _2048_H

#include <stdbool.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

#ifndef TABLE_SIZE
# define TABLE_SIZE 4
#endif //TABLE_SIZE

typedef enum { left, down, up, right, } Direction;

/* Call this function once you know the direction the user pressed */
void _2048_swipe             (int table[][TABLE_SIZE], Direction d);

void _2048_move              (int table[][TABLE_SIZE], int ref[][TABLE_SIZE],
void _2048_merge             (int table[][TABLE_SIZE], int ref[][TABLE_SIZE],
bool _2048_table_has_space   (int table[][TABLE_SIZE]);
bool _2048_table_has_changed(int table[][TABLE_SIZE], int old[][TABLE_SIZE]
void _2048_table_spawn       (int table[][TABLE_SIZE]);


#ifdef _2048_IMPLEMENTATION

int old[TABLE_SIZE][TABLE_SIZE];

void
_2048_swipe(int table[][TABLE_SIZE], Direction d)
{
    int x0, xn, dx;
    int y0, yn, dy;

    switch(d)
    {
        case left:
            x0 = 0           ; xn = +TABLE_SIZE; dx = +1;
```

```
            y0 = 1            ; yn = +TABLE_SIZE; dy = +1;
            break             ;

        case right:
            x0 = 0            ; xn = +TABLE_SIZE; dx = +1;
            y0 = TABLE_SIZE-2; yn = -1          ; dy = -1;
            break             ;

        case down:
            x0 = TABLE_SIZE-2; xn = -1          ; dx = -1;
            y0 = 0            ; yn = +TABLE_SIZE; dy = +1;
            break             ;

        case up:
            x0 = 1            ; xn = +TABLE_SIZE; dx = +1;
            y0 = 0            ; yn = +TABLE_SIZE; dy = +1;
            break;

        default: break; //unreachable
    }

    // we need this to check if the table has changed

    memcpy(old, table, TABLE_SIZE*TABLE_SIZE*sizeof(int));
    // ref is used to not merge a recently merged cell with another
    // as an example we can image we swipe right on the following:
    // * 4 4 2 2
    // we want the output to be
    // * 0 0 8 4
    // not
    // * 0 0 4 8
    int ref[TABLE_SIZE][TABLE_SIZE];

    for (int x = x0; (dx > 0 ? x < xn : x > xn); x += dx)
        for (int y = y0; (dy > 0 ? y < yn : y > yn); y += dy)
            if (table[x][y])
                _2048_merge(table, ref, x, y, d);
    memset(ref, 0, TABLE_SIZE*TABLE_SIZE*sizeof(int));

    for (int x = x0; (dx > 0 ? x < xn : x > xn); x += dx)
        for (int y = y0; (dy > 0 ? y < yn : y > yn); y += dy)
            if (table[x][y])
                _2048_move(table, ref, x, y, d);

    if (_2048_table_has_changed(old, table))
```

```
85          _2048_table_spawn(table);
86
87  }
88
89  void
90  _2048_merge(int table[][TABLE_SIZE], int ref[][TABLE_SIZE], int x, int y, D
91  {
92      int ox, oy; // offset
93      int ax, ay; // activate x y or k
94      int k0, kn, dk; // for(int k = k0; k < kn; k += dk)
95      switch(d)
96      {
97          case left:
98              ax =  1; ay =  0; ox = x  ;   oy = y-1;
99              k0 = y-1;  kn = -1         ;  dk = -1;
100             break;
101         case down:
102             ax =  0; ay =  1; ox = x+1;   oy = y  ;
103             k0 = x+1;  kn = +TABLE_SIZE;  dk = +1;
104             break;
105         case up:
106             ax =  0; ay =  1; ox = x-1;   oy = y  ;
107             k0 = x-1;  kn = -1         ;  dk = -1;
108             break;
109         case right:
110             ax =  1; ay =  0; ox = x  ;   oy = y+1;
111             k0 = y+1;  kn = +TABLE_SIZE;  dk = +1;
112             break;
113         default: break;
114     }
115
116     for (int k = k0; (dk > 0 ? k < kn: k > kn); k += dk)
117     {
118         int cx  = ax*ox + ay*k;
119         int cy  = ay*oy + ax*k;
120         if (table[cx][cy])
121         {
122             if (table[cx][cy] == table[x][y] && ref[cx][cy] != 1)
123             {
124                 table[cx][cy] += table[x][y];
125                 ref[cx][cy] = 1;
126                 table[x][y] = 0;
127                 return;
128             }
129             return;
```

```
130             }
131         }
132         return;
133     }
134
135     void
136     _2048_move(int table[][TABLE_SIZE], int ref[][TABLE_SIZE], int x, int y, Di
137     {
138         int ox, oy; // offset
139         int ax, ay; // activate x y or k
140         int k0, kn, dk; // for(int k = k0; k < kn; k += dk)
141         switch(d)
142         {
143             case left:
144                 ax =  1; ay =  0; ox = x  ; oy = y-1;
145                 k0 = y-1; kn = -1         ; dk = -1;
146                 break;
147             case down:
148                 ax =  0; ay =  1; ox = x+1; oy = y  ;
149                 k0 = x+1; kn = +TABLE_SIZE; dk = +1;
150                 break;
151             case up:
152                 ax =  0; ay =  1; ox = x-1; oy = y  ;
153                 k0 = x-1; kn = -1         ; dk = -1;
154                 break;
155             case right:
156                 ax =  1; ay =  0; ox = x  ; oy = y+1;
157                 k0 = y+1; kn = +TABLE_SIZE; dk = +1;
158                 break;
159             default: break;
160         }
161
162         for (int k = k0; (dk > 0 ? k < kn + 1 : k > kn - 1); k += dk)
163         {
164             int cx  = ax*ox + ay*k;
165             int cy  = ay*oy + ax*k;
166             int cxk = ax*ox + ay*(k-dk);
167             int cyk = ay*oy + ax*(k-dk);
168             /* if (obstacle) */
169             // obstacle may mean:
170             // 1. cell right after current cell
171             // 2. cell up until the last legal cell (including)
172             // 3. outside boundry cell (imaginary)
173             if (table[cx][cy])
174             {
```

```
175                 table[cxk][cyk] = table[x][y]; // 2
176                 if (k != k0) // 1
177                     table[x][y] = 0;
178                 return;
179             }
180             if (k == kn) // 3
181             {
182                 table[cxk][cyk] = table[x][y];
183                 table[x][y] = 0;
184                 return;
185             }
186         }
187         return;
188     }
189
190     bool
191     _2048_table_has_changed(int table[][TABLE_SIZE], int old[][TABLE_SIZE])
192     {
193         for (int i = 0; i < TABLE_SIZE; i++)
194             for (int j = 0; j < TABLE_SIZE; j++)
195                 if (table[i][j] != old[i][j])
196                     return true;
197         return false;
198     }
199
200     bool
201     _2048_table_has_space(int table[][TABLE_SIZE])
202     {
203         for (int i = 0; i < TABLE_SIZE; i++)
204             for (int j = 0; j < TABLE_SIZE; j++)
205                 if (table[i][j] == 0)
206                     return true;
207         return false;
208     }
209
210     void
211     _2048_table_spawn(int table[][TABLE_SIZE])
212     {
213         bool generated = false;
214         if (_2048_table_has_space(table))
215         {
216             while(!generated)
217             {
218                 int x = rand()%TABLE_SIZE;
219                 int y = rand()%TABLE_SIZE;
```

```
220                 if (!table[x][y])
221                 {
222                     int number;
223                     int r = rand();
224                     if (r % 10 < 9 )
225                         number = 2;
226                     else
227                         number = 4;
228                     table[x][y] = number;
229                     generated = true;
230                 }
231             }
232         }
233     }
234
235     #endif //_2048_IMPLEMENTATION
236
237     #endif // _2048_H
```

3.1.2 Main program

```
/*
 * Configuration
 */
#define TABLE_SIZE 4
#define _2048_IMPLEMENTATION
#include "2048.h"
int table[TABLE_SIZE][TABLE_SIZE];
typedef enum {
      flat = 0,
      l, d, u, r
} Orientation;

#include <FastLED.h>
#define LED_PIN 32
#define LED_BRIGHTNESS 16 // 0-255
#define LED_COUNT 64
CRGB leds[LED_COUNT];

#include <basicMPU6050.h>
basicMPU6050<> mpu;

#define seed() random()
void setup()
{
        srand(seed());
        _2048_table_spawn(table);
        _2048_table_spawn(table);

        FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds, LED_COUNT);
        FastLED.setBrightness(LED_BRIGHTNESS);

        mpu.setup();
        mpu.setBias();
}

/*
 * Logic
 */
CRGB cell_color(int x)
{
        if (x) return CHSV(7*x%360-30, 255, 255);
        else   return CHSV(0, 0, 0);
}
```

```
44
45  void draw_cell(int number, int x, int y)
46  {
47          CRGB color = cell_color(number);
48          leds[(2*x+0)*8 + (2*y+0)] = color;
49          leds[(2*x+0)*8 + (2*y+1)] = color;
50          leds[(2*x+1)*8 + (2*y+0)] = color;
51          leds[(2*x+1)*8 + (2*y+1)] = color;
52
53  }
54
55  void display(int table[][TABLE_SIZE])
56  {
57          for (int i = 0; i < TABLE_SIZE; i++)
58                  for (int j = 0; j < TABLE_SIZE; j++)
59                          draw_cell(table[i][j], i, j);
60          FastLED.show();
61  }
62
63  #define THRESHOLD 0.7
64  bool sub(float f) { return (-THRESHOLD < f && f < +THRESHOLD); }
65  bool sup(float f) { return (-THRESHOLD > f && f > +THRESHOLD); }
66
67  Orientation getOrientation()
68  {
69          Orientation o;
70          mpu.updateBias();
71          if (sup(mpu.ax()) && sub(mpu.ay()) && sub(mpu.az())) o = flat;
72          if (sub(mpu.az()) && mpu.ax() < -THRESHOLD) o = u;
73          if (sub(mpu.az()) && mpu.ay() < -THRESHOLD) o = l;
74          if (sub(mpu.az()) && mpu.ax() > +THRESHOLD) o = d;
75          if (sub(mpu.az()) && mpu.ay() > +THRESHOLD) o = r;
76          return o;
77  }
78
79  Orientation prev_orientation = flat;
80  void loop()
81  {
82          display(table);
83          Orientation curr_orientation = getOrientation();
84          if (curr_orientation != prev_orientation)
85                  switch (curr_orientation)
86                  {
87                          case l: _2048_swipe(table, left);  break;
88                          case d: _2048_swipe(table, down);  break;
```

```
89                          case u: _2048_swipe(table, up);    break;
90                          case r: _2048_swipe(table, right); break;
91                  }
92
93          prev_orientation = curr_orientation;
94  }
```

3.2 Code description

3.2.1 2048.h

`2048.h` is a single header library, where common functions for the 2048 game are immplemented, such as swipe, spawn, merge, has space, etc.

3.2.2 FastLED

Multi-platform library for controlling dozens of different types of LEDs along with optimized math, effect, and noise functions. `FastLED` is a fast, efficient, easy-to-use Arduino library for programming addressable LED strips and pixels such as WS2810, WS2811, LPD8806, Neopixel and more. FastLED also provides high-level math functions that can be used for generative art and graphics. [https://docs.arduino.cc/libraries/fastled/]

3.2.3 basicMPU6050

lightweight library for the MPU6050. library to configure and retrieve the raw sensor outputs of the MPU6050. It includes simples routines to calibrate the gyro. [https://docs.arduino.cc/libraries/basicmpu6050/]

3.2.4 setup

The `setup` function initializes the three essential components of the software.

3.2.5 display

The `display()` function is designed to render the game table on the LED matrix. It utilizes the `draw_cell()` function, which is responsible for drawing individual cells on the LED matrix. The color of each cell can be configured using the `cell_color()` function.

3.2.6 input

The `getOrientation()` function determines the orientation of the sensor. Its sensitivity can be adjusted by modifying the value of the `THRESHOLD` macro.

3.2.7 loop

The `loop()` function serves as the main event loop of the program, coordinating the execution of all other functions. It updates the display, retrieves the current orientation, and updates the game state using the swipe functions from the `2048.h` library.

4 Testing

4.1 Configuration

4.1.1 2048.h

The first component included is the game logic, which resides in the file *2048.h*. As *2048.h* is a single-header library, the `_2048_IMPLEMENTATION` macro must be defined to ensure that the library includes the implementations of the game functions, rather than just their declarations.

After this, a declaration of the game table is made, which will store the state of the game. The direction enumeration type from *2048.h* is extended to include the **flat** position and is renamed to `Orientation` accordingly.

4.1.2 FastLED.h

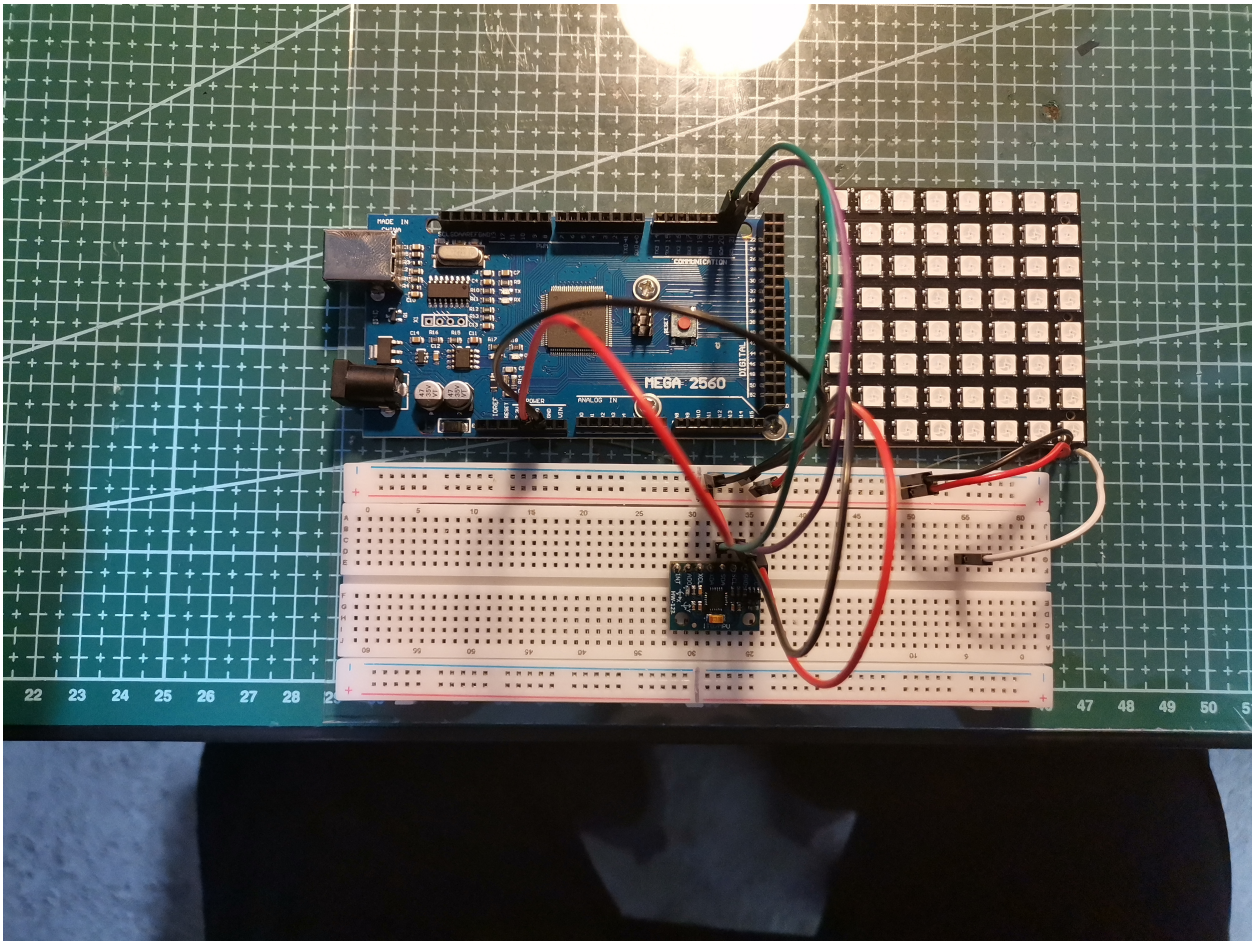
The next library included is *FastLED.h*, which allows communication with the display. The `LED_PIN` is configured and assigned to pin `32`; this can be changed to any other port as needed. The `LED_BRIGHTNESS` ranges from `0-255`, and an arbitrary value of `16` has been chosen, but it can be adjusted as required. The `LED_COUNT` is set to `64`, reflecting the number of LEDs in the display. Following this, the `leds` array is defined with a length matching `LED_COUNT`.

4.1.3 basicMPU6050.h

The next library, *basicMPU6050.h*, makes it easy to communicate with the MPU6050 sensor. An object named `mpu` is created to handle this interaction.

4.1.4 Setup

In the `setup` function, the random number generator (RNG) is initialized. The 2048 game table is set up with two tiles. The display and gyroscope are then configured.



4.2 Logic

4.2.1 Display

The `display()` function processes each row and column of the game table, calling the `draw_cell()` function for each cell. Once completed, it calls `FastLED.show()` to refresh the display. The `draw_cell()` function assigns colors based on the current values in the game table to show on the display. Additionally, a function named `cell_color` is used to return a color based on the input number, using a predefined formula that can be adapted.

4.2.2 Input

The `getOrientation()` function retrieves inclination information from the accelerometer by calling the methods `mpu.ax()`, `mpu.ay()`, and `mpu.az()`. It compares these values against a configurable `THRESHOLD`. The function utilizes two helper methods, `sup()` and `sub()`, which verify whether a value falls within the specified `THRESHOLD`.

4.2.3 Loop

The first action carried out by the `loop()` function is to display the current state of the game table. It then obtains the current orientation and compares it with the previous orientation. If a difference is found, the function determines the current orientation using a `switch` statement and calls the appropriate function from `2048.h`. Finally, it updates the `prev_orientation` variable.