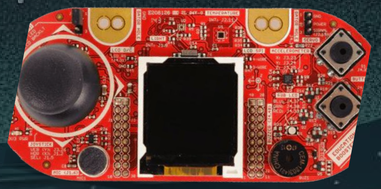
Rocket Game

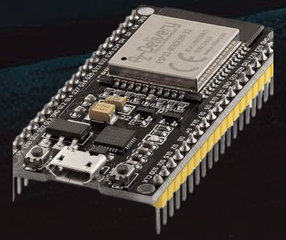
Konstantinos Zefkilis, Filippo Maffei, Federico Iop, Luca Pio Pierno

Introduction

- We created a game where a rocket must avoid obstacles.
- The goal is to emulate a classic game from old phones or arcade machines on the Booster Pack screen.
- The player moves the rocket left or right using the Booster Pack's joystick.
- Avoid red square obstacles and stay on black squares to collect points.
- Added visual and audible feedback:
  - Green LED while playing.
  - Red LED and "Game Over" sound when losing.



- The project uses BoosterPackMKII, MSP432, and ESP32 interconnected.



- ESP32 retrieves the top global scores, saved on the server.

Representative code (screen update)

In order to have a screen that updates quickly, we implemented the updateGrid() function. These are the most important parts of the function:

```
for (row = GRID_HEIGHT - 2; row > 0; row--) {
    for (col = 0; col < GRID_WIDTH; col++) {
        grid[row][col] = grid[row - 1][col];
    }
}
```

Copies the squares of the previous row to the current row. So that the grid is updated.

```
for (row = 0; row < GRID_HEIGHT; row++) {
    for (col = 0; col < GRID_WIDTH; col++) {
        //Code here
    }
}
```

Iterate each array element, then colour it black (free space), red (obstacle) or green (rocket)

```
int newCubeColumn = rand() % GRID_WIDTH;
for (col = 0; col < GRID_WIDTH; col++) {
    if (col == newCubeColumn) {
        grid[0][col] = 1; // red cube on first row
    } else {
        grid[0][col] = 0; // black cube
    }
}
```

Initially it generates a random number in which to place the red square (obstacle) and assigns the value 1 (obstacle) or 0 (free) to each element of the array represented by the grid

Representative code (screen update)

```
if (grid[row][col] == 0 || row == GRID_HEIGHT-1){
    Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_BLACK);
    Graphics_Rectangle rectangle = {col * 16, row * 16, col * 32, row * 32};
    Graphics_fillRectangle(&g_sContext, &rectangle);
} else {
    Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_RED);
    Graphics_Rectangle rectangle = {0, 0,  16, 16};
    Graphics_fillRectangle(&g_sContext, &rectangle);
}
if(grid[row][col] == 2){
    Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_YELLOW);
    Graphics_Rectangle rectangle = {rocketPos * 16, row * 16, rocketPos * 32, row * 32};
    Graphics_fillRectangle(&g_sContext, &rectangle);
}
```

```
Grid Values:
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 2 0 0 0 0
```

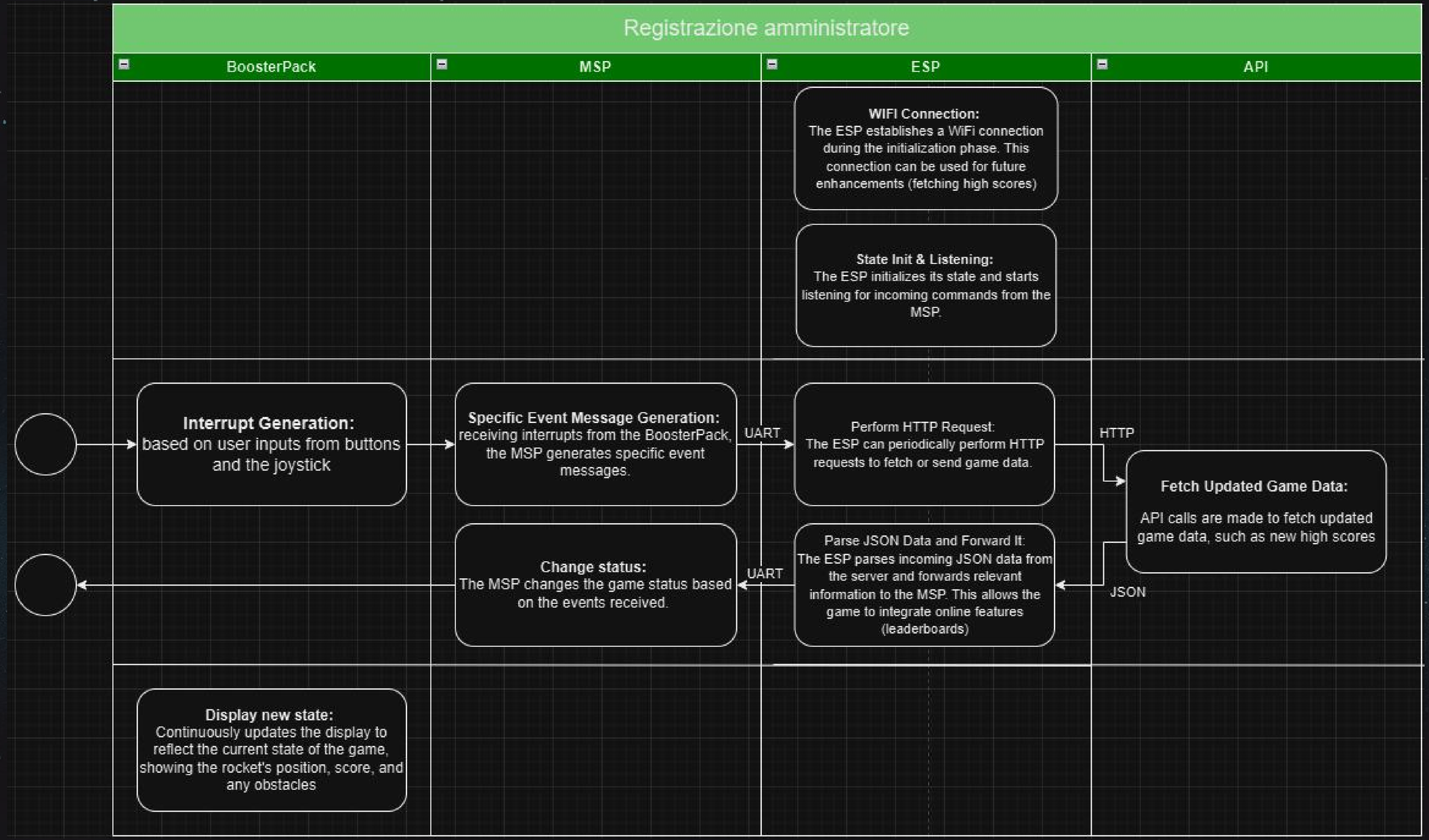
If the current array element is 0 (free square) or it is the last row (GRID_HEIGHT - 1) which must be all black except the rocket square it will be coloured black, otherwise it will be coloured red. If, on the other hand, the value of the current array element is 2, the square will be coloured green (rocket).

Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_BLACK); //All graphic operations will use the specified colour (black)

Graphics_Rectangle rectangle = {0, 0, 16, 16}; // Defines a rectangle with respective angle coordinates

Graphics_fillRectangle(&g_sContext, &rectangle); // Fills the square with the previously chosen colour

Working Flow



Wi-fi Test & UART Test

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to Wi-Fi...");
}
Serial.println("Connected");
```

This test is essential to verify that the device can successfully connect to a Wi-Fi network. It ensures that the device is ready to communicate with other devices or servers via the network.

```
void setup() {
  Serial.begin(baud_rate);
  Serial2.begin(baud_rate, SERIAL_8N1, RX, TX);
}

if (Serial2.available() > 0) {
  String command = Serial2.readStringUntil('\n');
  Serial.println("Received command: " + command);
}
```

This test verifies the device's ability to communicate via the UART. It ensures that data can be correctly received by another device and displayed for debugging.

Future features

- **Improve graphics:** Implement advanced textures and smooth animations to improve the visual appearance of the game.
- **Use accelerometer to move the rocket:** Use the device's accelerometer to control the rocket's movements, providing a more intuitive and immersive gaming experience.
- **Receive more information through ESP with other controls:** Integrate ESP modules to receive external data, allowing new features and advanced controls to be added.

Qr-code with the video