Department of Computer Science
Technical University of Cluj-Napoca

Design with Microprocessors

Semester project: Battleships

14 January 2021

Name: Tudor-Stefan Todea
Group: 30434
Email: todeastudor@gmail.com
14 January 2021

Teaching Assistant: Attila Füzes

Contents

Chapter 1

Project Description

The aim of this project is to build a functional 2-player game of Battleships using several Arduino Uno boards. Each player has a grid of 6 by 6 buttons, each of them representing a position on the enemy player board. Each player presses a button to select which position they wish to attack. If there is a ship in that attacked position, an LED will light up in the corresponding position in that player's LED matrix and the buzzer will play a sound. If the player misses, no LED will light up and the buzzer will play a different sound. Each player has the possibility of resetting their board using the button on the Arduino board connected to the LEDs. The game ends when a player has hit three enemy ships, meaning that they have 6 LEDs lit up on their board (since every ship spawns over two positions). The project can be easily scaled further, but since it was build using Tinkercad, which is just a simulator that uses JavaScript at its core, we are unable to do so.

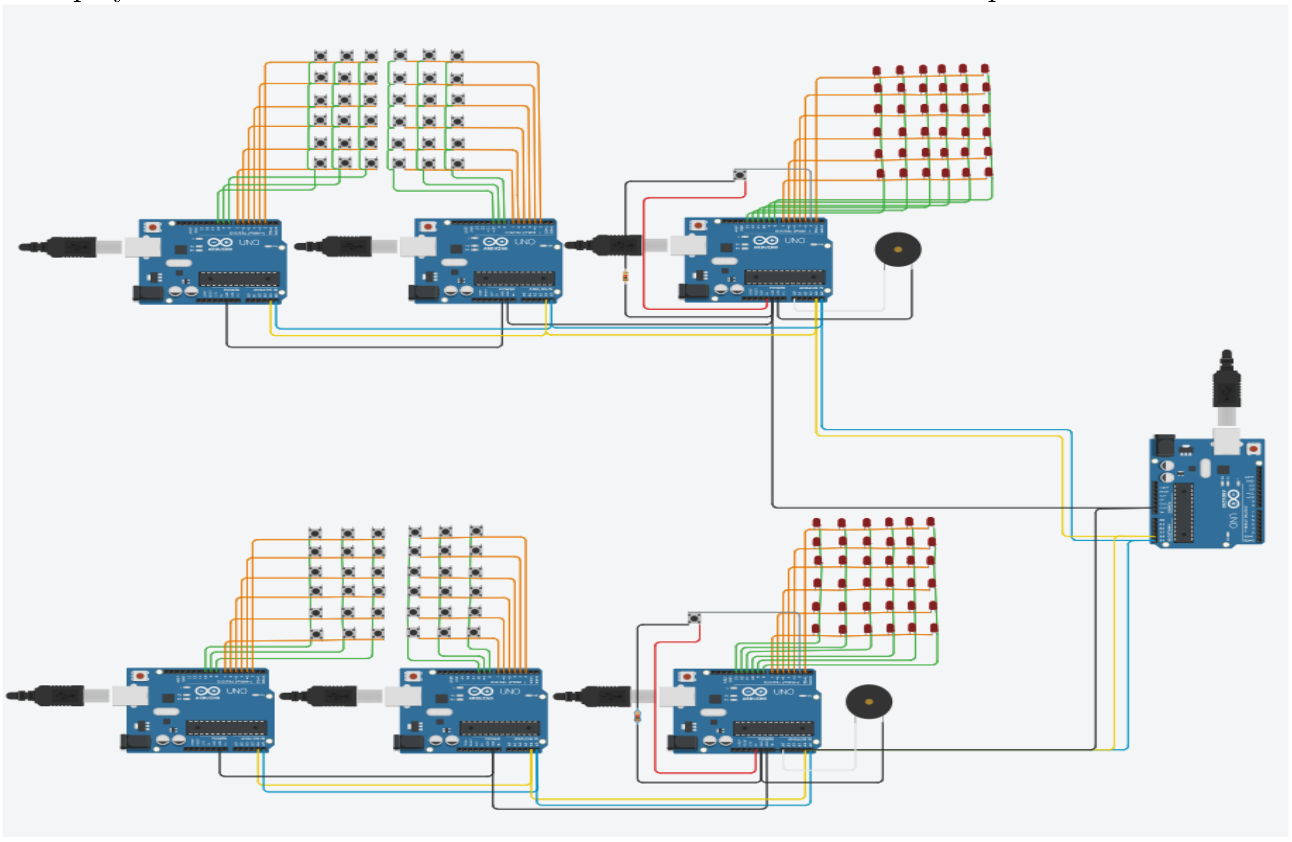
Chapter 2

Components

The components used in the project were the following:

- 7 Arduino Uno boards (3 for each player and 1 as the referee)
- 74 Buttons (37 for each player, out of which 36 are for attacking and one for resetting the board)
- 72 Red LEDs (36 for each player)
- 2 Piezo Buzzers (one for each player)
- 2 1000 Ω Resistors (for the reset button of each player)
- Wires

All of the 7 boards are connected through I2C, each player having 3 boards. On the first two boards of the players we have 36 buttons (18 on each board) which transmit the button pressed to the third board, each button having a code from 0 to 35. The third board has the reset button connected to it and the 36 LEDs displaying the state of the game for that player. The board in the middle acts as a master-reader and keeps track of the turns.



Chapter 3

Implementation

3.1 Player 1 Boards

Button Board 1

```
#include <Wire.h>

#define ROWS_COUNT 6
#define COLS_COUNT 3

#define TIME_DEBOUNCE 6

int CHARACTERS[ROWS_COUNT][COLS_COUNT] =

{
  {3,4,5},
  {9,10,11},
  {15,16,17},
  {21,22,23},
  {27,28,29},
  {33,34,35},
};

int ROWS[ROWS_COUNT] = {2,3,4,5,6,7}, COLS[COLS_COUNT] = {10,9,8};

void MATRIX_BUTTON_PRESSED(int R, int C){
  Wire.beginTransmission(9);
  Wire.write(CHARACTERS[R][C]);
  Wire.endTransmission();


}



void setup(){

  for(int R = 0; R < ROWS_COUNT; R++){

    pinMode(ROWS[R],INPUT_PULLUP);
  }
  for(int C = 0; C < COLS_COUNT; C++){
    pinMode(COLS[C],OUTPUT);
    digitalWrite(COLS[C],1);
  }

  Wire.begin();


}


void loop(){

  for(int C = 0; C < COLS_COUNT; C++){

    for(int R = 0; R < ROWS_COUNT; R++){

      digitalWrite(COLS[C],0);
      if(!(digitalRead(ROWS[R]))) MATRIX_BUTTON_PRESSED(R,C);
      digitalWrite(COLS[C],1);
      delay(TIME_DEBOUNCE);
    }

  }

}
```

Button Board 2

```
#include <Wire.h>

#define ROWS_COUNT 6
#define COLS_COUNT 3

#define TIME_DEBOUNCE 6

int CHARACTERS[ROWS_COUNT][COLS_COUNT] =

{
  {0,1,2},
  {6,7,8},
  {12,13,14},
  {18,19,20},
  {24,25,26},
  {30,31,32},
};

```

```
int ROWS[ROWS_COUNT] = {2,3,4,5,6,7}, COLS[COLS_COUNT] = {10,9,8};

void MATRIX_BUTTON_PRESSED(int R, int C){
  Wire.beginTransmission(9);
  Wire.write(CHARACTERS[R][C]);
  Wire.endTransmission();


}


void setup(){

  for(int R = 0; R < ROWS_COUNT; R++){

    pinMode(ROWS[R],INPUT_PULLUP);
  }
  for(int C = 0; C < COLS_COUNT; C++){
    pinMode(COLS[C],OUTPUT);
    digitalWrite(COLS[C],1);
  }

  Wire.begin();


}


void loop(){

  for(int C = 0; C < COLS_COUNT; C++){

    for(int R = 0; R < ROWS_COUNT; R++){

      digitalWrite(COLS[C],0);
      if(!(digitalRead(ROWS[R]))) MATRIX_BUTTON_PRESSED(R,C);
      digitalWrite(COLS[C],1);
      delay(TIME_DEBOUNCE);
    }

  }

}
```

Player 1 Main Board

This board acts as a slave-writer. It can receive data from the referee regarding the turn state and from the button boards regarding the pressed button. When it is the first player's turn, we wait for a button to be pressed. We check if the code of the button corresponds to a hit or miss, by checking if there is a ship in that position (a '1' in the BattleShips position matrix). If an enemy ship is hit, we light up the LED of that respective position, play a sound and increment the number of positions hit. When 6 positions are hit, the game is over and player one wins. After the first player takes his turn, the turn variable turns to '2', thus signaling the referee that it needs to pass the control to the second player.

```
#include <Wire.h>
int countHits = 0;
char turn = '1';
bool p1Turn = true;
int LedMatrix[6][6] =
  {
    {1,1,1,1,1,1},
    {1,1,1,1,1,1},
    {1,1,1,1,1,1},
    {1,1,1,1,1,1},
    {1,1,1,1,1,1},
    {1,1,1,1,1,1}
  };

int BattleshipPositions[6][6] =
  {
    {0,0,0,0,0,0},
    {0,0,0,0,1,1},
    {1,1,0,0,0,0},
    {0,0,0,0,0,0},
    {0,0,0,0,0,1},
    {0,0,0,0,0,1}
  };

int ledRows[6]={1,3,4,5,6,7};
int ledCols[6]={13,12,11,10,9,8};
void setup() {


  Wire.begin(9);
  Wire.onReceive(receiveEvent);
  Wire.onRequest(requestEvent);
  Serial.begin(9600);
  for (int i = 2; i < 14; i++){
    pinMode(i,OUTPUT);
    digitalWrite(i,LOW);
  }
  pinMode(2, INPUT);
  EIMSK |= (1 << INT0);
  EICRA |= (1 << ISC01);
  sei();
}

ISR(INT0_vect)
{
  countHits = 0;
  for (int i = 0; i < 6; i++) {
    for (int j = 0; j < 6; j++) {
      BattleshipPositions[i][j] = 0;
        LedMatrix[i][j] = 1;
    }
  }
}

void receiveEvent(int bytes) {
  int x = Wire.read();
  if (x == '0' + 1) {
     p1Turn = true;
     turn = '1';
   }
  if ((p1Turn == true) && (x <= 35)) {
    int row = x / 6;
    int col = x % 6;
    if (BattleshipPositions[row][col] == 1) {
      if (LedMatrix[row][col] == 1) {
        LedMatrix[row][col] = 0;
        countHits++;
      }
      if (countHits > 5) {
        tone(A0, 2400, 125);
        tone(A0, 2400, 125);
        tone(A0, 2400, 125);
        tone(A0, 2400, 125);
        tone(A0, 2400, 125);
      }else {
        tone(A0, 1500, 125);
      }
    }else {
      tone(A0, 73, 125);
    }
    turn = '2';
    p1Turn = false;
    Serial.println(x);
  }

}

void requestEvent() {
  Wire.write(turn);

}
void loop() {
  for(int c = 0; c < 6; c++){
    digitalWrite(ledCols[c],HIGH);
    for(int r = 0; r < 6; r++){
      digitalWrite(ledRows[r],1000*LedMatrix[r][c]);
    }
    for (int r = 0; r < 6; r++){
      digitalWrite(ledRows[r], HIGH);
     delay(1);
    }
   digitalWrite(ledCols[c],LOW);
  }
}
```

3.2 Referee Board

The referee is a master-reader, who continuously (every 0.1 seconds) requests the value of the turn variable of each player. When the value sent by the two boards differs, meaning it is time to change turns, we check the transmitted character from each board in the previous iteration, to see if we need to pass the turn from the first player to the second player or vice-versa.

```
#include "Wire.h"
char prevReadCharFromP1;
char prevReadCharFromP2;
void setup() {

  Wire.begin();
  Serial.begin(9600);
}

void loop()
{
  Wire.requestFrom(9, 1);
  char readCharFromP1 = Wire.read();
  Wire.requestFrom(10, 1);
  char readCharFromP2 = Wire.read();
  if (readCharFromP1 != readCharFromP2) {
    if (readCharFromP1 == '2' && readCharFromP2 == '1') {
      if (prevReadCharFromP1 == '1') {
        Wire.beginTransmission(10);
      Wire.write('2');
      Wire.endTransmission();
      }
      if (prevReadCharFromP2 == '2') {
        Wire.beginTransmission(9);
      Wire.write('1');
      Wire.endTransmission();
      }

    }
  }
  prevReadCharFromP1 = readCharFromP1;
  prevReadCharFromP2 = readCharFromP2;
 delay(100);
}
```

The code for the boards used by the second player is similar to the code used for the boards of the first player. The whole project can be found in this GitHub repository, while the full Tinkercad simulation can be accessed here.