Mastermind Electronic Edition

Final Project Report

Name: Michal Bochenek

Matric No: 40270585

# Abstract:

1. The main reason behind this topic choice was to recreate a gaming experience of an old childhood board game called Mastermind using electronic elements (Arduino and Android smartphone) in which one player creates a secret pin code and the other is trying to guess it in limited number of moves.
2. Each game starts with a randomly chosen sequence of 3 pins of 5 different colours, a player will use smartphone application to pick the sequence and send it to Arduino the application is programmed to check the sequence against the hidden one and show the chosen answer by lighting 3 multi-coloured LED lights of corresponding colours and 3 yellow and 3 red pins showing tips of the hidden password.
3. Deliverables:  
   - A Smartphone application (the game) written with the use of Processing

* A working Bluetooth connection between the involved devices
* An Arduino device with the premade LED light circuit board and Bluetooth connection with the above application.

**Declaration of authorship: I confirm that the work submitted is mine and that wherever possible the work of others has been clearly acknowledged and referenced.**

**Signature: Michal Bochenek Date: 03/04/2017**

Table of Contents

[Abstract: 1](#_Toc478998197)

[1. Introduction. 3](#_Toc478998198)

[1.1 Background: 3](#_Toc478998199)

[1.2 Aims and deliverables: 3](#_Toc478998200)

[2. Design choices for the Mastermind Electronic Edition game: 4](#_Toc478998201)

[2.1. Introduction: 4](#_Toc478998202)

[2.2. Design section: 4](#_Toc478998203)

[2.2.1. Graphical user interface: 4](#_Toc478998204)

[2.2.2. Arduino circuit board design: 6](#_Toc478998205)

[3. Implementation of the design: 7](#_Toc478998206)

[4. Interface test plan: 9](#_Toc478998207)

[5. Results evidence: 12](#_Toc478998208)

[6. Conclusions and further progress propositions: 14](#_Toc478998209)

[7. Self-evaluation: 14](#_Toc478998210)

[8. References: 15](#_Toc478998211)

[9. Appendices: 15](#_Toc478998212)

[a. Appendix A: System Software Code Listing – Android 15](#_Toc478998213)

[b. Appendix B: System Software Code Listing – Arduino 34](#_Toc478998214)

[c. Appendix C: Hardware used 40](#_Toc478998215)

[d. Appendix D: Test Results: 40](#_Toc478998216)

[e. Appendix E: Initial Project Proposal 47](#_Toc478998217)

[f. Appendix F: Interim report 50](#_Toc478998218)

Table of figures:

1. Figure 1: Mastermind original game board
2. Figure 2: Player interface final design
3. Figure 3: Circuit board design
4. Figure 4: Code listing example for Android
5. Figure 5: Code listing example for Arduino
6. Figures 6 through 9: Evidence of the application interface working

# 1. Introduction.

## Background:

A Mastermind is a game in which one player sets a secret code and the other player is trying to guess the sequence by placing the coloured pins in the holes, with each attempt (usually 12 attempts per game) the player is given feedback on how many colours were guessed right and how many of pins are placed in the right position. The game usually uses 4 pin holes and 6 colours.

The proposed solution will differ from the original in such way that the one player is replaced by the randomizing the pin sequence and the colour sequence is being assembled in the application, then sent to the Arduino to show the results (the alternative will be using the application to determine the result to display on the device instead).

To make things run easier and due to the limitation of the Arduino device – the player will have less pins to choose from yet also less attempts to guess the code.

## Aims and deliverables:

Aim of this project is to enhance student knowledge and practical application of the already possessed skills gained during the subject classes.

The project will intend to deliver:

- Working game – smartphone application.

- Arduino device connecting over Bluetooth with the game app.

- Any required documentation containing project technical aspects, testing, code listing etc.

# Design choices for the Mastermind Electronic Edition game:

## Introduction:

The design and the rules of the game were based purely on the original Mastermind board game created by Mordecai Meirowitz in 70’s [1] without changing the idea a player is given the board – in this case an application containing an user interface which communicates itself with the Arduino to create the representation of the player pin choices.

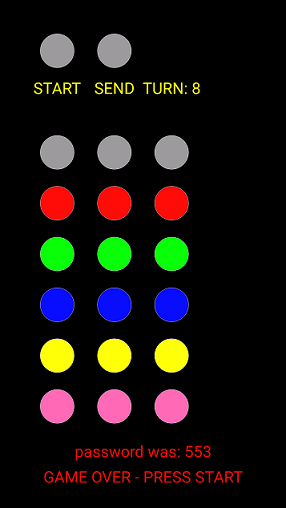
Each time the player is awarded yellow pins led if the right colour was selected in the sequence, except the case where the pin is also occupying the right position, then the player is awarded with red pin instead, guessing all 3 pins lights 3 red lights and is a signal that the game is now over. Player has to guess the answer in 8 attempts.  
[fig.1]

## Design section:

## Graphical user interface:

The initial user interface was replaced with new one, the reason for this was that the selected library - G4Ptool version 4.2 was not meeting its requirements, mainly it was not working in Arduino mode, which was not clearly stated by the author of the content.  
To work around this problem a manual coding approach was taken – student created the required content himself based on tutorial for Processing [2].

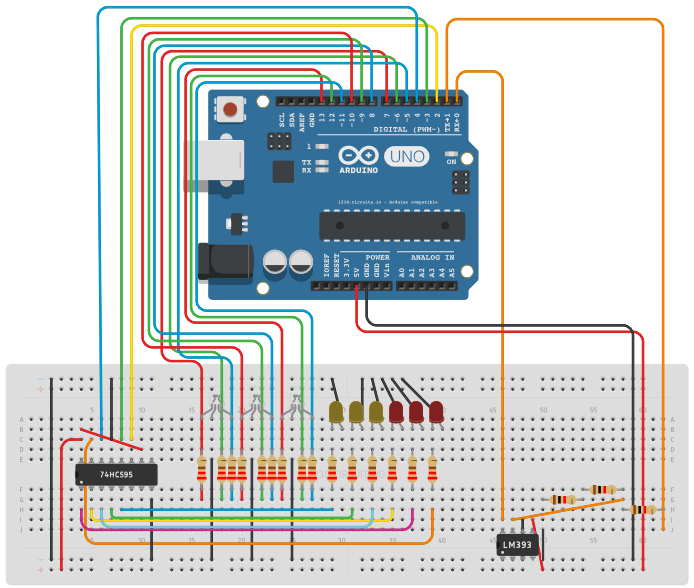
The interface was then set up and redesigned so that in the end it looked as shown on the picture below:

[fig.2]

At the upper left side there is a start button and send data button and the turn counter. Pressing start resets the counter and creates a new password. To actually start a new game the player must press start at least once.  
Pressing send button changes the Bluetooth’s broadcasted data signal.  
In the middle of the screen each of the three columns of circles represents a single pin – the coloured pins are the colour choices for given password pin – the player taps the screen to select the colour which is then updated and replaces the grey pin.

## Arduino circuit board design:

The circuit board design was not altered since the early project implementation, only missing ground connection was added at the voltage divider as shown: [fig.3]



# Implementation of the design:

The only external library used with this project was Ketai – for Bluetooth connection management [3].  
For coding the Android only Processing language was used and the Arduino code was written using its own language and IDE.  
As an example the game rules were realized in such way that the player has no possibility to look at the correct password during the game, the values are passed within the game logics and are stored there, the interface in the application has input gathering purpose only and displaying important output messages:  
[fig.4]

|  |
| --- |
| /\*  this should return either 0, 1, 2 or 3 for use with yellow LED  each right colour can be awarded only once and only if it was not  occupying the right position  \*/  public int checkRightColours(int val1, int val2, int val3){  int colours = 0;  //award player with either 2 (different colours) or 1 yellow pins:  if(val1==getPin1()){  if(val2==getPin3()&&val3==getPin2()){  return 2;  }  else if(val2==getPin3()||val3==getPin2()){  return 1;  }  }  if(val2==getPin2()){  if(val1==getPin3()&&val3==getPin1()){  return 2;  }  else if(val1==getPin3()||val3==getPin1()){  return 1;  }  }  if(val3==getPin3()){  if(val1==getPin2()&&val2==getPin1()){  return 2;  }  else if(val1==getPin2()||val2==getPin1()){  return 1;  }  }  //check if valid for remaining pins, best case it will return 3  if(val1!=getPin1()&&(val1==getPin2()||val1==getPin3())){  colours++;  }  if(val2!=getPin2()&&(val2==getPin1()||val2==getPin3())){  colours++;  }  if(val3!=getPin3()&&(val3==getPin1()||val3==getPin2())){  colours++;  }  return colours;  } |

Arduino code implementation was really simple, yet very efficient, it was based on the student’s code written at the previous classes and extended with additional methods for controlling the LED lights. [fig.5]

|  |
| --- |
| void loop(){  //if there is a bt signal:  if(BTserial.available()){  //String st = "";  int i = 0;  while(BTserial.available()){  //read a single bt byte token at a time:  c = BTserial.read();  //cast c to int and assign it to a pos array  pos[i]=(int)c;  //st = st+pos[i]+" ";  //Serial.println("position: "+st);  i++;  }  //here what does the program need to do before reading new bt signal:  lightLeds();  leds = (int)pos[3];  //Serial.println(leds);  updateShiftRegister(leds);  delay(200);  }  } |

# Interface test plan:

Test plan for interface testing template is attached with this document.

**1. Scope of the test:**

This tests purpose is to check if the prototype is fully functional, thus it has to perform well under project specification requirements – in this case the game must be playable.

**2. Output example for a single test:**

A successful test will deliver an output that is an evidence of a prototype working in compliance with the project i.e. display GUI buttons or react to the tap and light selected colours on Arduino.

**3. Instructions to the tester:**

- use Processing version 3 or newer [4]

- locate the “MastermindPrj” folder and open it within Processing

- compile the code and run the prototype application

- use Arduino IDE version 1.8.1 [5]

- navigate and open within the Arduino IDE project folder named “projectMasterMind”

- connect the prepared Arduino circuit board, compile the code (if it is first use) and run the serial console

- follow the test cases

**4. Test risks / issues:**

This test has one main issue – it can’t be used for a proper in-code debugging as it is a black box test type, an external tester doesn’t know the code and may need to refer to the listing in order to check for the expected result.

**5. Items to be tested/not tested:**

The actual test result tables and test cases will be attached as an appendix.  
Items to test: ready application, Arduino circuit board and interface.  
Items to be not tested: the code.

**6. Test approach:**

To perform a test, tester has to follow up the test cases and compare the expected results with actual if they provide correct output, please refer to the game rules specification if in doubt.

**7. Test pass/ fail criteria:**

Test passes if the tested item expected result is met with an actual output. Failure is when there is any other or no output evidence shown.

**8. Test entry / exit criteria:**

Entry criteria:

- Testable code is available, but assume that it is not presented to the tester

- Test data is checked, test case scenarios are given to the tester

- Test data used is correct and an up to date version

Exit criteria:

- Verify if all tests were run and documented

- Verify if tests covered the area of research valid for the test approach

- Verify the activities were completed on schedule

**9. Test schedule:**

Test has to be completed at least once before releasing the prototype for the final submission.

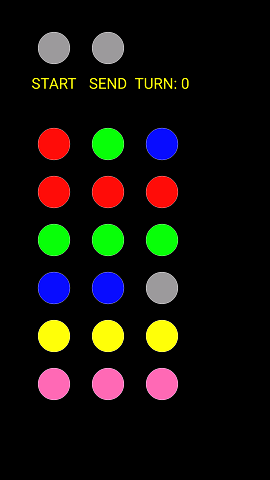
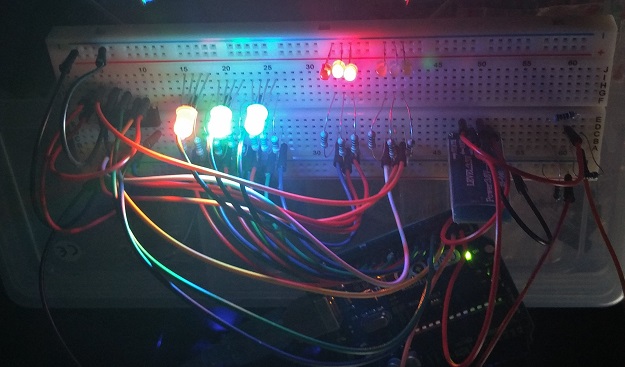
Any errors found should be removed or a proper commentary should be given, and prototype has to be re-tested.

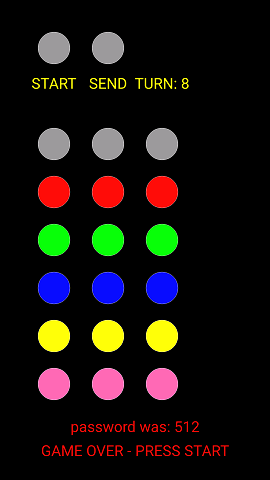
**10. Test environmental/ staffing / training needs:**

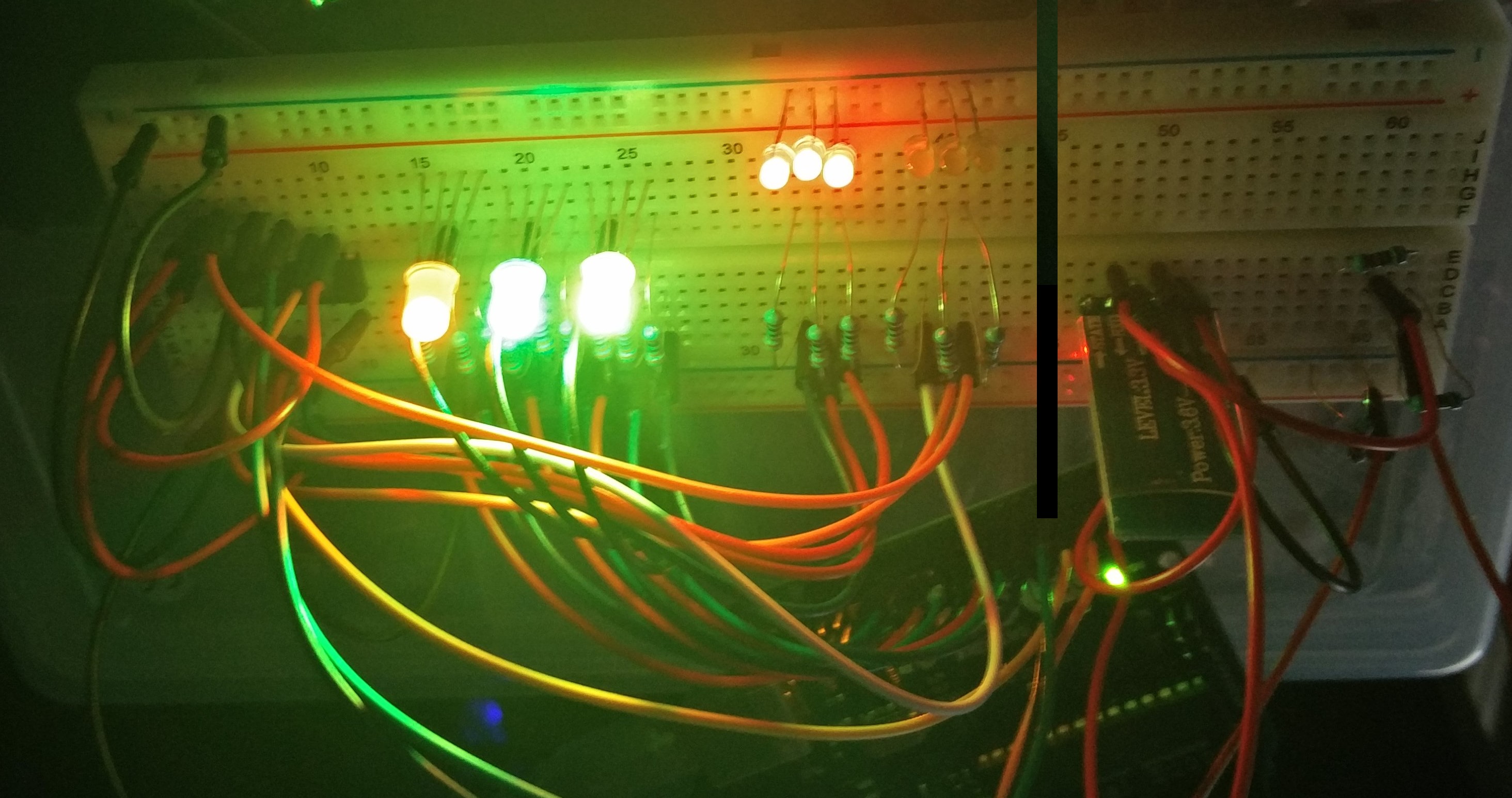
Test should be performed in closed, black box environment, tester can’t have an access to the code listing, the test case scenarios are given to him/her at the place and project specification must be available as testers might need to compare the expected result with an actual output produced.

At least one or more testers are required with good knowledge of Arduino and basic knowledge of Android environment, and project specification, also a basic knowledge of black box testing is essential.

# Results evidence:

**Interface working – reacting to player input taps:**  
Player input red, green and blue pin (above) [fig.6]  
**Arduino reacting to the player choices:**   
Guessed 2 Red pins in 1st move (above) – Red, Green and Blue LEDs are shown [fig.7]

**Game can be lost or won:**  
Game lost (above)[fig.8]Game won(above)[fig.9]



# Conclusions and further progress propositions:

The project was successful and the whole process was running smoothly and without great obstacles. The game met the requirements of the original Mastermind board game and it is in playable state.  
  
The Bluetooth communication between the devices is based on a simple yet very useful byte array containing just 4 positions which are broadcasted and received at the Arduino side which then processes these numbers, 3 positions are coding the led colour to be turned on and 4th position contains a value for the shift register which is then converted into binary and lights the right red and yellow led.   
  
In future the number of pins can be easily expanded by adding additional shift registers on circuit board and the interface has each pin represented as an object so adding more pins to handle requires just a minor code and app layout adjustments.  
Android GUI could be also expanded by additional data, graphical effects or even sound/vibration.

There was an attempt taken by the student to try to write Arduino part of the code as a proto-thread (creating multiple separate threads to be handled by the Arduino single chip) where Bluetooth communications, shift register and multi-LED lights were running separately but it was abandoned due to the high issue with time synchronization although it was working, yet a flickering effect was very visible.

# Self-evaluation:

The strong sides of this project was that it wasn’t trying to invent a wheel but rather implement existing solution which is well known but as a board game and give it a nice redesign using electronic components and the smartphone as an user interface providing device.   
In the whole process the student can enhance his/hers abilities to communicate the different devices working over different languages using wireless “technology bridge” between them knowing what data structure is being sent over the radio waves. In this case it was Bluetooth but other technologies can be easily used like Wi-Fi or even infrared.  
This project also improved greatly the knowledge gained during the course and provided a solid background for future attempts as a hobby.

# References:

[1]"Mastermind (board game)", *En.wikipedia.org*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Mastermind\_(board\_game). [Accessed: 03- Apr- 2017].

[2]"Processing.js", *Processingjs.org*, 2017. [Online]. Available: http://processingjs.org/learning/topic/buttons/. [Accessed: 03- Apr- 2017].

[3] "Ketai Library - Download", *Ketai.org*, 2017. [Online]. Available: http://ketai.org/download/. [Accessed: 03- Apr- 2017].

[4] "Download \ Processing.org", *Processing.org*, 2017. [Online]. Available: https://processing.org/download/?processing. [Accessed: 03- Apr- 2017].

[5]"Arduino - Software", *Arduino.cc*, 2017. [Online]. Available: https://www.arduino.cc/en/Main/Software. [Accessed: 03- Apr- 2017].

# Appendices:

## Appendix A: System Software Code Listing – Android

**Main Class – MastermindPrj:**  
//Created on: 12/03/2017 by 40270585

//version status: final

//main class for the game, responsible for: event handling, GUI drawing, bt connection,

//game setup and running the program

import android.content.Intent;

import android.os.Bundle;

import ketai.net.bluetooth.\*;

import ketai.ui.\*;

import ketai.net.\*;

//setup app start up config and bt

PFont f;

boolean bReleased = true; //no permament sending when finger is tap

KetaiBluetooth bt;

GameLogics ng = new GameLogics();

boolean isConfiguring = true;

String info = "";

KetaiList klist;

ArrayList devicesDiscovered = new ArrayList();

byte[] data = {0, 0, 0, 0};

//setup game:

boolean locked = false;

boolean sending = false;

boolean gameRunning = false;

boolean gameEnd = false;

int redPins = 0;

int yellowPins = 0;

int turn = 0;

//color variables

color currentcolor;

//base colours:

color red = color(255,12,12);

color green = color(12, 255, 12);

color blue = color(12, 12, 255);

//mixed colours:

color yellow = color(255,255,12);

color pink = color(255,105,180);

//default circle starting colour:

color startGrey = color(155, 155, 155);

color basecolor = color(200, 100, 100);

color highlight = color(155,155,155);

//create non button pins:

Circle circle1, circle2, circle3;

//create buttons:

CircleButton sendButton, startButton;

CircleButton circle1red, circle2red, circle3red;

CircleButton circle1blue, circle2blue, circle3blue;

CircleButton circle1green, circle2green, circle3green;

CircleButton circle1yellow, circle2yellow, circle3yellow;

CircleButton circle1pink, circle2pink, circle3pink;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The following code is required to enable bluetooth at startup.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void onCreate(Bundle savedInstanceState){

super.onCreate(savedInstanceState);

bt = new KetaiBluetooth(this);

}

void onActivityResult(int requestCode, int resultCode, Intent data){

bt.onActivityResult(requestCode, resultCode, data);

}

void setup(){

//Get device display size

size(displayWidth, displayHeight);

//setup graphics ui:

smooth();

frameRate(15);

orientation(PORTRAIT);

//start listening for BT connections

bt.start();

//at app start select device…

isConfiguring = true;

//font config

f = createFont("SansSerif",30,true);

//draw password pins - don't refresh them, colours set only by player events:

circle1 = new Circle(displayWidth\*0.2, displayHeight\*0.3);

circle2 = new Circle(displayWidth\*0.4, displayHeight\*0.3);

circle3 = new Circle(displayWidth\*0.6, displayHeight\*0.3);

}

void draw() {

//PREPARE GUI:

//set black background

background(0);

//button start new game:

textFont(f);

textAlign(CENTER);

fill(yellow);

textSize(60);

text("START", displayWidth\*0.2, displayHeight\*0.185);

startButton = new CircleButton(displayWidth\*0.2, displayHeight\*0.1, 128, startGrey, highlight);

//button send

textFont(f);

textAlign(CENTER);

fill(yellow);

textSize(60);

text("SEND", displayWidth\*0.4, displayHeight\*0.185);

sendButton = new CircleButton(displayWidth\*0.4, displayHeight\*0.1, 128, startGrey, highlight);

//turn counter:

textFont(f);

textAlign(CENTER);

fill(yellow);

textSize(60);

text("TURN: "+turn, displayWidth\*0.6, displayHeight\*0.185);

//for debug only show red and yellow pins:

//text("Yellow: "+yellowPins, displayWidth\*0.2+500, displayHeight\*0.1+200);

//fill(red);

//text("Red: "+redPins, displayWidth\*0.2+500, displayHeight\*0.1+300);

//set game over text:

if(turn == 8){

gameRunning = false;

fill(red);

text("password was: "+ng.getPin1()+ng.getPin2()+ng.getPin3(), displayWidth\*0.5, displayHeight\*0.9);

fill(red);

textAlign(CENTER);

text("GAME OVER - PRESS START", displayWidth\*0.5, displayHeight\*0.95);

}

//button led 1:

circle1red = new CircleButton(displayWidth\*0.2, displayHeight\*0.4, 128, red, highlight);

circle1green = new CircleButton(displayWidth\*0.2, displayHeight\*0.5, 128, green, highlight);

circle1blue = new CircleButton(displayWidth\*0.2, displayHeight\*0.6, 128, blue, highlight);

circle1yellow = new CircleButton(displayWidth\*0.2, displayHeight\*0.7, 128, yellow, highlight);

circle1pink = new CircleButton(displayWidth\*0.2, displayHeight\*0.8, 128, pink, highlight);

//button led 2:

circle2red = new CircleButton(displayWidth\*0.4, displayHeight\*0.4, 128, red, highlight);

circle2green = new CircleButton(displayWidth\*0.4, displayHeight\*0.5, 128, green, highlight);

circle2blue = new CircleButton(displayWidth\*0.4, displayHeight\*0.6, 128, blue, highlight);

circle2yellow = new CircleButton(displayWidth\*0.4, displayHeight\*0.7, 128, yellow, highlight);

circle2pink = new CircleButton(displayWidth\*0.4, displayHeight\*0.8, 128, pink, highlight);

//button led 3:

circle3red = new CircleButton(displayWidth\*0.6, displayHeight\*0.4, 128, red, highlight);

circle3green = new CircleButton(displayWidth\*0.6, displayHeight\*0.5, 128, green, highlight);

circle3blue = new CircleButton(displayWidth\*0.6, displayHeight\*0.6, 128, blue, highlight);

circle3yellow = new CircleButton(displayWidth\*0.6, displayHeight\*0.7, 128, yellow, highlight);

circle3pink = new CircleButton(displayWidth\*0.6, displayHeight\*0.8, 128, pink, highlight);

//at app start - select device using Ketai interface:

if (isConfiguring) {

ArrayList names;

background(78, 93, 75);

klist = new KetaiList(this, bt.getPairedDeviceNames());

isConfiguring = false;

}

//once bt device is selected create GUI:

else{

update(mouseX, mouseY);

sendButton.display();

startButton.display();

circle1.display();

circle1red.display();

circle1green.display();

circle1blue.display();

circle1yellow.display();

circle1pink.display();

circle2.display();

circle2red.display();

circle2green.display();

circle2blue.display();

circle2yellow.display();

circle2pink.display();

circle3.display();

circle3red.display();

circle3green.display();

circle3blue.display();

circle3yellow.display();

circle3pink.display();

if(gameRunning==false){

//send a byte signal array to BT:

bt.broadcast(data);

}

else{

//send a byte signal array to BT:

bt.broadcast(data);

//System.out.println("game started"+" "+data[0]+" "+data[1]+" "+data[2]+" "+data[3]);

}

}

}

void update(int x, int y){

circle1.update();

circle2.update();

circle3.update();

if(locked == false){

circle1red.update();

circle1green.update();

circle1blue.update();

circle1yellow.update();

circle1pink.update();

circle2red.update();

circle2green.update();

circle2blue.update();

circle2yellow.update();

circle2pink.update();

circle3red.update();

circle3green.update();

circle3blue.update();

circle3yellow.update();

circle3pink.update();

sendButton.update();

startButton.update();

}

else{

locked = false;

}

if(mousePressed){

if(circle1red.pressed()) {

circle1.setColor(red);

circle1.setValue(1);

}

else if(circle1green.pressed()) {

circle1.setColor(green);

circle1.setValue(2);

}

else if(circle1blue.pressed()) {

circle1.setColor(blue);

circle1.setValue(3);

}

else if(circle1yellow.pressed()) {

circle1.setColor(yellow);

circle1.setValue(4);

}

else if(circle1pink.pressed()) {

circle1.setColor(pink);

circle1.setValue(5);

}

else if(circle2red.pressed()) {

circle2.setColor(red);

circle2.setValue(1);

}

else if(circle2green.pressed()) {

circle2.setColor(green);

circle2.setValue(2);

}

else if(circle2blue.pressed()) {

circle2.setColor(blue);

circle2.setValue(3);

}

else if(circle2yellow.pressed()) {

circle2.setColor(yellow);

circle2.setValue(4);

}

else if(circle2pink.pressed()) {

circle2.setColor(pink);

circle2.setValue(5);

}

else if(circle3red.pressed()) {

circle3.setColor(red);

circle3.setValue(1);

}

else if(circle3green.pressed()) {

circle3.setColor(green);

circle3.setValue(2);

}

else if(circle3blue.pressed()) {

circle3.setColor(blue);

circle3.setValue(3);

}

else if(circle3yellow.pressed()) {

circle3.setColor(yellow);

circle3.setValue(4);

}

else if(circle3pink.pressed()) {

circle3.setColor(pink);

circle3.setValue(5);

}

else if(sendButton.pressed()) {

//if fields are grey or the game is already sending data the button wont send anything:

if(sending == false && circle1.getValue()!=0&& circle2.getValue()!=0&& circle3.getValue()!=0)

{

sending = true;

//check if the game should end now:

gameEnd = ng.checkGameEnd(circle1.getValue(),circle2.getValue(),circle3.getValue());

//check red pins:

redPins = ng.checkRightPosition(circle1.getValue(),circle2.getValue(),circle3.getValue());

//can't get yellow pins if this is true:

if(redPins >= 2){

yellowPins=0;

}

else{

yellowPins = ng.checkRightColours(circle1.getValue(),circle2.getValue(),circle3.getValue());

}

//send with BT: {led1, led2, led3, red leds, yellow leds}

data[0] = (byte)circle1.getValue();

data[1] = (byte)circle2.getValue();

data[2] = (byte)circle3.getValue();

data[3] = (byte)ng.setShiftReg(redPins, yellowPins);

//this is to indicate if the game has ended on android:

if(gameEnd==false)

{

circle1.setColor(startGrey);

circle2.setColor(startGrey);

circle3.setColor(startGrey);

turn++;

}

else{

//you can now start a new game also broadcast to arduino will be stopped from now on:

gameRunning = false;

//System.out.println("game finished");

}

//end of button pressed evt:

sending = false;

}

}

else if(startButton.pressed() && gameRunning==false) {

gameRunning = ng.startGame();

ng.reroll();

//System.out.println("game started!");

turn = 0;

}

else{

//handle nothing selected in future

}

}

}

void onKetaiListSelection(KetaiList klist){

String selection = klist.getSelection();

bt.connectToDeviceByName(selection);

//dispose of list for now

klist = null;

}

//Call back method to manage data received

void onBluetoothDataEvent(String who, byte[] data){

if (isConfiguring)

return;

//received

info += new String(data);

//clean if string to long

if(info.length() > 150)

info = "";

}

// Arduino+Bluetooth+Processing

// Arduino-Android Bluetooth communication

**Button Class:**

//Created on: 12/03/2017 by 40270585

//code was based on the example from:

//http://processingjs.org/learning/topic/buttons/

//this class provides Button object type to be used for button events

class Button{

float x, y;

float size;

color basecolor, highlightcolor;

color currentcolor;

boolean over = false;

boolean pressed = false;

void update(){

if(over()){

currentcolor = highlightcolor;

}

else{

currentcolor = basecolor;

}

}

boolean pressed(){

if(over){

locked = true;

return true;

}

else{

locked = false;

return false;

}

}

boolean over(){

return true;

}

boolean overCircle(float x, float y, float diameter){

float disX = x - mouseX;

float disY = y - mouseY;

if(sqrt(sq(disX) + sq(disY)) < diameter/2 ){

return true;

}

else {

return false;

}

}

}

**Circle Class:**

//Created on: 12/03/2017 by 40270585

//code was based on the example from:

//http://processingjs.org/learning/topic/buttons/

//its responsible for handling inactive cirlce buttons

class Circle{

float x;

float y;

float size = 128;

int value = 0;

color background = color(155,155,155);

Circle(float ix, float iy){

x = ix;

y = iy;

}

void display(){

stroke(255);

fill(getColor());

ellipse(x, y, size, size);

}

void update(){

this.background = getColor();

}

void setColor(color selected){

this.background = selected;

}

color getColor(){

return background;

}

int getValue(){

return value;

}

void setValue(int v){

value = v;

}

}

**Circle Button Class:**

//Created on: 12/03/2017 by 40270585

//code was based on the example from:

//http://processingjs.org/learning/topic/buttons/

//it's responsible for handling active circle buttons

class CircleButton extends Button{

CircleButton(float ix, float iy, float isize, color icolor, color ihighlight){

x = ix;

y = iy;

size = isize;

basecolor = icolor;

highlightcolor = ihighlight;

currentcolor = basecolor;

}

boolean over(){

if( overCircle(x, y, size) ){

over = true;

return true;

}

else {

over = false;

return false;

}

}

void display(){

stroke(255);

fill(currentcolor);

ellipse(x, y, size, size);

}

void setColor(color selected){

stroke(255);

fill(selected);

ellipse(this.x, this.y, this.size, this.size);

}

}

**Game Logics Class:**

//Created on: 12/03/2017 by 40270585

//version status: final

//this is the game logics class for handling the game rules

class GameLogics{

//create variabes to hold player pin colour choice

int pin1=0;

int pin2=0;

int pin3=0;

//construct a new game and generate password:

private GameLogics(){

pin1 = int(random(1,6));

pin2 = int(random(1,6));

pin3 = int(random(1,6));

}

//ensure that the game was set to start

public boolean startGame(){

return true;

}

//GETTERS:

public int getPin1(){

return pin1;

}

public int getPin2(){

return pin2;

}

public int getPin3(){

return pin3;

}

//check if the game is finished [password was found]:

public boolean checkGameEnd(int val1, int val2, int val3){

if(pin1==val1&&pin2==val2&&pin3==val3){

//game should end

return true;

}

else{

return false;

}

}

//this should return either 0, 1, 2 or 3 for use with red LED

public int checkRightPosition(int val1, int val2, int val3){

int pins =0;

//for each value check:

if(val1==getPin1()){

pins++;

}

if(val2==getPin2()){

pins++;

}

if(val3==getPin3()){

pins++;

}

return pins;

}

/\*

this should return either 0, 1, 2 or 3 for use with yellow LED

each right colour can be awarded only once and only if it was not

occupying the right position

\*/

public int checkRightColours(int val1, int val2, int val3){

int colours = 0;

//award player with either 2 (different colours) or 1 yellow pins:

if(val1==getPin1()){

if(val2==getPin3()&&val3==getPin2()){

return 2;

}

else if(val2==getPin3()||val3==getPin2()){

return 1;

}

}

if(val2==getPin2()){

if(val1==getPin3()&&val3==getPin1()){

return 2;

}

else if(val1==getPin3()||val3==getPin1()){

return 1;

}

}

if(val3==getPin3()){

if(val1==getPin2()&&val2==getPin1()){

return 2;

}

else if(val1==getPin2()||val2==getPin1()){

return 1;

}

}

//check if valid for remaining pins, best case it will return 3

if(val1!=getPin1()&&(val1==getPin2()||val1==getPin3())){

colours++;

}

if(val2!=getPin2()&&(val2==getPin1()||val2==getPin3())){

colours++;

}

if(val3!=getPin3()&&(val3==getPin1()||val3==getPin2())){

colours++;

}

return colours;

}

//This method returns a value to be transfered on Arduino

//Arduino then reads it and sends to shift register to light up

//the proper leds - int a - red pins - int b- yellow pins

int setShiftReg(int a, int b){

if(a == 0 && b == 1){

return 1;

}

if(a == 0 && b == 2){

return 3;

}

if(a == 0 && b == 3){

return 7;

}

if(a == 1 && b == 0){

return 8;

}

if(a == 1 && b == 1){

return 9;

}

if(a == 1 && b == 2){

return 11;

}

if(a == 2 && b == 0){

return 24;

}

if(a == 3 && b == 0){

return 56;

}

return 0;

}

//reroll the random pin numbers method:

public void reroll(){

pin1 = int(random(1,6));

pin2 = int(random(1,6));

pin3 = int(random(1,6));

}

}

## Appendix B: System Software Code Listing – Arduino

**Main Class:**

//Author: 40270585

//Date: 15/03/2017 version final

//This is the main code for the Arduino Mastermind Electric Edition project

#include <SoftwareSerial.h>

//SHIFT REGISTER:

//Pin connected to ST\_CP of 74HC595 (green wire)

int latchPin = 3;

//Pin connected to SH\_CP of 74HC595 (yellow wire)

int clockPin = 2;

////Pin connected to DS of 74HC595(blue wire)

int dataPin = 4;

//3-coloured LEDs:

//1

int redPin = 13;

int greenPin = 12;

int bluePin = 11;

//2

int redPin2 = 10;

int greenPin2 = 9;

int bluePin2 = 8;

//3

int redPin3 = 7;

int greenPin3 = 6;

int bluePin3 = 5;

//BLUETOOTH:

SoftwareSerial BTserial(0, 1); // RX | TX

// Connect the HC-05 TX to Arduino pin 2 RX.

// Connect the HC-05 RX to Arduino pin 3 TX through a voltage divider.

//Incoming bluetooth signal as an ascii character [8bits]:

char c = ' ';

int leds = 0;

byte num = 0;

int pos[] = {0,0,0,0};

//set the led colour based on given value, using values lower than 255

//reduces possible flickering

void setColor(int pinR, int pinG, int pinB, int value){

if(value==0)

{

analogWrite(pinR, 0);

analogWrite(pinG, 0);

analogWrite(pinB, 0);

}

//red

else if(value==1){

analogWrite(pinR, 224);

analogWrite(pinG, 0);

analogWrite(pinB, 0);

}

//green

else if(value==2){

analogWrite(pinR, 0);

analogWrite(pinG, 224);

analogWrite(pinB, 0);

}

//blue

else if(value==3){

analogWrite(pinR, 0);

analogWrite(pinG, 0);

analogWrite(pinB, 224);

}

//yellow

else if(value==4){

analogWrite(pinR, 224);

analogWrite(pinG, 224);

analogWrite(pinB, 0);

}

//pink

else if(value==5){

analogWrite(pinR, 224);

analogWrite(pinG, 100);

analogWrite(pinB, 172);

}

else{}

}

//switch the leds on

void lightLeds(){

setColor(redPin, greenPin, bluePin, pos[0]);

setColor(redPin2, greenPin2, bluePin2, pos[1]);

setColor(redPin3, greenPin3, bluePin3, pos[2]);

}

void updateShiftRegister(int s){

// the LEDs don't change while you're sending in bits:

digitalWrite(latchPin, LOW);

delay(5);

// shift out the bits:

shiftOut(dataPin, clockPin, MSBFIRST, s);

//take the latch pin high so the LEDs will light up:

digitalWrite(latchPin, HIGH);

// pause before next value:

delay(5);

}

//void blinkAll\_2Bytes() {

// digitalWrite(latchPin, 0);

// shiftOut(dataPin, clockPin, 0);

// shiftOut(dataPin, clockPin, 0);

// digitalWrite(latchPin, 1);

// delay(200);

// for (int x = 0; x < 10; x++) {

// digitalWrite(latchPin, 0);

// shiftOut(dataPin, clockPin, 255);

// shiftOut(dataPin, clockPin, 255);

// digitalWrite(latchPin, 1);

// delay(20);

// digitalWrite(latchPin, 0);

// shiftOut(dataPin, clockPin, 0);

// shiftOut(dataPin, clockPin, 0);

// digitalWrite(latchPin, 1);

// delay(20);

// }

//}

void setup() {

//3C LEDs:

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

pinMode(bluePin, OUTPUT);

pinMode(redPin2, OUTPUT);

pinMode(greenPin2, OUTPUT);

pinMode(bluePin2, OUTPUT);

pinMode(redPin3, OUTPUT);

pinMode(greenPin3, OUTPUT);

pinMode(bluePin3, OUTPUT);

//SHIFT REGISTER

pinMode(latchPin, OUTPUT);

pinMode(clockPin, OUTPUT);

pinMode(dataPin, OUTPUT);

//BLUETOOTH

Serial.begin(9600);

Serial.println("Arduino is ready");

Serial.println("Remember to select Both NL & CR in the serial monitor");

// HC-05 default serial speed for AT mode is 38400

BTserial.begin(9600);

}

void loop(){

//if there is a bt signal:

if(BTserial.available()){

//String st = "";

int i = 0;

while(BTserial.available()){

//read a single bt byte token at a time:

c = BTserial.read();

//cast c to int and assign it to a pos array

pos[i]=(int)c;

//st = st+pos[i]+" ";

//Serial.println("position: "+st);

i++;

}

//here what does the program need to do before reading new bt signal:

lightLeds();

leds = (int)pos[3];

//Serial.println(leds);

updateShiftRegister(leds);

delay(200);

}

}

## Appendix C: Hardware used

Arduino Uno board   
USB cable for power supply  
32 cables – medium and long length recommended  
3 Yellow led lights   
3 Red led lights  
3 Multi-coloured led lights  
15 330 Ohm Resistors  
3 1kOhm Resistors  
Bluetooth module – HC-05 (ZS 040)  
8bit Shift register – SN74HC595  
OnePlus2 Smartphone (for the app testing) – with android 6.0.1  
USB 3.0 cable – for smartphone connection with the PC  
PC – development platform – any PC capable to run Processing and Arduino IDE

## Appendix D: Test Results:

Test plan:

|  |  |
| --- | --- |
| **Test 1 : Application Start** | Last run date: |
| Purpose:  To test the application correct start  Method:  Turn on the application on the testing mobile device  Project Files Used: Test Filename(s):  MastermindPrj Folder  Expected Result:  The application will ask for the Bluetooth access on the device and display initial menu ( device selector)  Outcome: {Pass or description of failure condition} | |

|  |  |
| --- | --- |
| **Test 2 : Establishing inter-device connection** | Last run date: |
| Purpose:  To test that Bluetooth on both devices creates a connection.  Method:  Turn on the application on the testing mobile device, switch Arduino on, follow instructions on mobile phone screen, turn the Arduino serial console on.  Project Files Used: Test Filename(s):  MastermindPrj Folder, projectMasterMind Folder  Expected Result:  Bluetooth on Arduino will connect with the device, the fast blinking pattern will become slower.  Outcome: {Pass or description of failure condition} | |

|  |  |
| --- | --- |
| **Test 3 : Game start – first turn** | Last run date: |
| Purpose:  To test the game has started.  Method:  Turn on the application on the testing mobile device, establish connection, press start button and take the first turn, then press send button.  Project Files Used: Test Filename(s):  MastermindPrj Folder, projectMasterMind Folder  Expected Result:  The Arduino device will show selected leds and light any red/yellow pins if appropriate, the turn counter will increment by 1 on the screen.  Outcome: {Pass or description of failure condition} | |

|  |  |
| --- | --- |
| **Test 4 : Game won/lost** | Last run date: |
| Purpose:  To test the game has ended properly.  Method:  Turn on the application on the testing mobile device, establish connection, press start button and take turns until either 3 red pins are lighted or turn counter will show turn 8 and game over message.  Project Files Used: Test Filename(s):  MastermindPrj Folder, projectMasterMind Folder  Expected Result:  The Arduino device will show selected leds and light any red/yellow pins if appropriate, the turn counter will increment by 1 on the screen until or before game over. The game will be lost or won depending on the gameplay.  Outcome: {Pass or description of failure condition} | |

|  |  |
| --- | --- |
| **Test 5 : Application exit/reconnection** | Last run date: |
| Purpose:  To test the application was turned off and reconnected.  Method:  Turn on the application on the testing mobile device, establish connection, press start button and take the first turn, then press send button. After a while close the application and check if the game was finished by turning the app again.  Project Files Used: Test Filename(s):  MastermindPrj Folder, projectMasterMind Folder  Expected Result:  The Arduino device will continue to run even if the application was shut, and if reconnected the game will start again with new password.  Outcome: {Pass or description of failure condition} | |

Test run:

|  |  |
| --- | --- |
| **Test 1 : Application Start** | Last run date: 03/04/17 |
| Purpose:  To test the application correct start  Method:  Turn on the application on the testing mobile device  Project Files Used: Test Filename(s):  MastermindPrj Folder  Expected Result:  The application will ask for the Bluetooth access on the device and display initial menu ( device selector)  Outcome: PASS, no issues found, the application detects no connection or missing device | |

|  |  |
| --- | --- |
| **Test 2 : Establishing inter-device connection** | Last run date: 03/04/17 |
| Purpose:  To test that Bluetooth on both devices creates a connection.  Method:  Turn on the application on the testing mobile device, switch Arduino on, follow instructions on mobile phone screen, turn the Arduino serial console on.  Project Files Used: Test Filename(s):  MastermindPrj Folder, projectMasterMind Folder  Expected Result:  Bluetooth on Arduino will connect with the device, the fast blinking pattern will become slower.  Outcome: PASS, the Bluetooth connection was established | |

|  |  |
| --- | --- |
| **Test 3 : Game start – first turn** | Last run date: 03/04/17 |
| Purpose:  To test the game has started.  Method:  Turn on the application on the testing mobile device, establish connection, press start button and take the first turn, then press send button.  Project Files Used: Test Filename(s):  MastermindPrj Folder, projectMasterMind Folder  Expected Result:  The Arduino device will show selected leds and light any red/yellow pins if appropriate, the turn counter will increment by 1 on the screen.  Outcome: PASSED, no issues found, turn incremented as seen of fig. below | |

|  |  |
| --- | --- |
| **Test 4 : Game won/lost** | Last run date: 03/04/17 |
| Purpose:  To test the game has ended properly.  Method:  Turn on the application on the testing mobile device, establish connection, press start button and take turns until either 3 red pins are lighted or turn counter will show turn 8 and game over message.  Project Files Used: Test Filename(s):  MastermindPrj Folder, projectMasterMind Folder  Expected Result:  The Arduino device will show selected leds and light any red/yellow pins if appropriate, the turn counter will increment by 1 on the screen until or before game over. The game will be lost or won depending on the gameplay.  Outcome: PASSED, game has ended, message was shown on fig. below: | |

|  |  |
| --- | --- |
| **Test 5 : Application exit/reconnection** | Last run date: 03/04/17 |
| Purpose:  To test the application was turned off and reconnected.  Method:  Turn on the application on the testing mobile device, establish connection, press start button and take the first turn, then press send button. After a while close the application and check if the game was finished by turning the app again.  Project Files Used: Test Filename(s):  MastermindPrj Folder, projectMasterMind Folder  Expected Result:  The Arduino device will continue to run even if the application was shut, and if reconnected the game will start again with new password.  Outcome: PASSED, android handles application shutdown properly, after restart a new game begins, Arduino also responds to the new input. | |

## Appendix E: Initial Project Proposal

# Mastermind Electronic Edition

**Initial Project Proposal**

Name: Michal Bochenek

Matric No: 40270585

Abstract

1. The main reason for this project topic choice is to re-create an old childhood board game called Mastermind in which player is trying to guess a hidden sequence of colours in limited amount of moves.
2. Each game will have randomly chosen sequence of colours, depending on the constraints it will be from 3 to up to 6 colours, a player will use smartphone application to create the sequence and send it to Arduino either the application or the device will be programmed to check the sequence against the hidden one and show the right answer by lighting LED lights of corresponding colour or white LED lights to show how many colours from the sequence player has selected right.
3. Deliverables:  
   - smartphone application (the game) written with the use of Processing

* Arduino device with premade LED light system and Bluetooth connection with the above application.

**Declaration of authorship: I confirm that the work submitted is mine and that wherever possible the work of others has been clearly acknowledged and referenced.**

**Signature : Michal Bochenek Date: 15/02/2017**

## Introduction

### Background/context

A Mastermind is a game in which one player sets a secret code and the other player is trying to guess the sequence by placing the coloured pins in the holes, with each attempt (usually 12 attempts per game) the player is given feedback on how many colours were guessed right and how many of pins are placed in the right position. The game usually uses 4 pin holes and 6 colours. The proposed solution will differ from the original in such way that the one player is replaced by the randomizing the pin sequence and the colour sequence is being assembled in the application, then sent to the Arduino to show the results (the alternative will be using the application to determine the result to display on the device instead).  
To make things run easier and due to limitation of the multi-coloured LED player will have less colours to choose from yet also less attempts to guess the code.

### Aims and deliverables

Aim of this project is to enhance student knowledge and practical application of the already possessed skills gained during the subject classes.  
The project will intend to deliver:   
- working game – smartphone application  
- Arduino device connecting over Bluetooth with the game app  
- any required documentation containing project technical aspects etc.

Design choices

Recreated Mastermind game with some changes to the rules mentioned in paragraph 1. Possible outcomes are such as the device will use:  
- row of 3-5 LED red lights to show the amount of pins placed in the right position,   
- row of 3-5 white LED lights to display the amount of pins with the right colour,   
- row of 3-5 multi-coloured LED lights displaying the end result as a reward to the player or once the player loses the game,  
- game application with the basic graphical user interface allowing user to select the sequence and send it over to Arduino,   
- outcomes will be strongly influenced by the skill and time limitations of the student

Method

Items required to deliver the project:  
- coloured LED lights   
- cables  
- breadboard(s)  
- Arduino Uno or similar, powered via USB cable or batteries  
- resistors  
- Analog multiplexer corresponding with the number of lights used  
- Android Smartphone with built in Bluetooth  
- Bluetooth transmitter/receiver for Arduino (receiver at least)  
  
  
Technology requirements:   
- Arduino programming development environment  
- Processing to create the Android application, it was discovered that processing can handle Arduino connection with proper expansions so that it can handle both GUI side and communication with the Arduino

Timeline:

|  |  |  |
| --- | --- | --- |
| Date | Deliverable | Comment |
| 17/02/2017 | Project initial documentation |  |
| 24/02/2017 | Project device scheme, and application GUI draft |  |
| 03/03/2017 | Project initial device and GUI working prototypes |  |
| 10/03/2017 | Catch up work |  |
| 17/03/2017 | Development of the application and first release for testing | Testing against the usability and project outputs |
| 24/03/2017 | Any other catch up work if needed | Can continue if time constraint will allow to. |

References

[1]"Arduino Playground - Processing", *Playground.arduino.cc*, 2017. [Online]. Available: http://playground.arduino.cc/Interfacing/Processing. [Accessed: 15- Feb- 2017].

[2]"Connecting Arduino to Processing - learn.sparkfun.com", *Learn.sparkfun.com*, 2017. [Online]. Available: https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing. [Accessed: 15- Feb- 2017].

[3]"Mastermind (board game)", *En.wikipedia.org*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Mastermind\_(board\_game). [Accessed: 15- Feb- 2017].

[4]P. Goldsborough, "Arduino and Multiplexing", *Thecodeinn.blogspot.co.uk*, 2017. [Online]. Available: http://thecodeinn.blogspot.co.uk/2013/10/arduino-and-multiplexing.html. [Accessed: 15- Feb- 2017].

## Appendix F: Interim report

# Title: Mastermind Electronic Edition

**Interim report**

Name: Michal Bochenek

Matric No: 40270585

Introduction

A Mastermind is a game in which one player sets a secret code and the other player is trying to guess the sequence by placing the coloured pins in the holes, with each attempt (usually 12 attempts per game) the player is given feedback on how many colours were guessed right and how many pins were placed in the right position.   
The game usually uses 4 pin holes and 6 colours. The proposed solution will differ from the original in such a way that a player who creates the secret code is replaced by the random pin sequence at the game start, the game starts once the player connects via Bluetooth to the Arduino device.   
To make things run easier for the prototype and due to the pin number the player will have less colours to choose from yet also less attempts to guess the code.

Aim of this project is to enhance student knowledge and practical application of the already possessed skills gained during the subject classes.  
The project will intend to deliver:   
- working game – a smartphone application  
- an Arduino device connected over Bluetooth with the game app  
- any required documentation containing project technical aspects etc.

Current status

The research during analysis wasn’t broad since all of the requirements for the successful delivery of the project was based on using existing skills gained from the previous classes, such as smartphone app GUI development, Bluetooth communication between two paired devices, how does shift register work, RGB LED lights colour pins check-up, and also the necessary electronic parts were analysed and they are as such:  
  
Items required to deliver the project:

- 3 x RGB LED lights

- 3 x Yellow LED lights

- 3 x Red LED lights

- Connection cables

- A large breadboard

- Arduino Uno or similar, powered via USB cable or batteries

- 15 x 330 Ohm Resistors

- 3 x 1k Ohm Resistors to create voltage divider (2:1 ratio)

- 74HC595 - 8-bit serial-in, serial or parallel-out Shift Register

- Android Smartphone with built in Bluetooth

- Bluetooth transmitter/receiver for Arduino (receiver at least)

Technological requirements:

- Arduino version 1.8.1 programming development environment

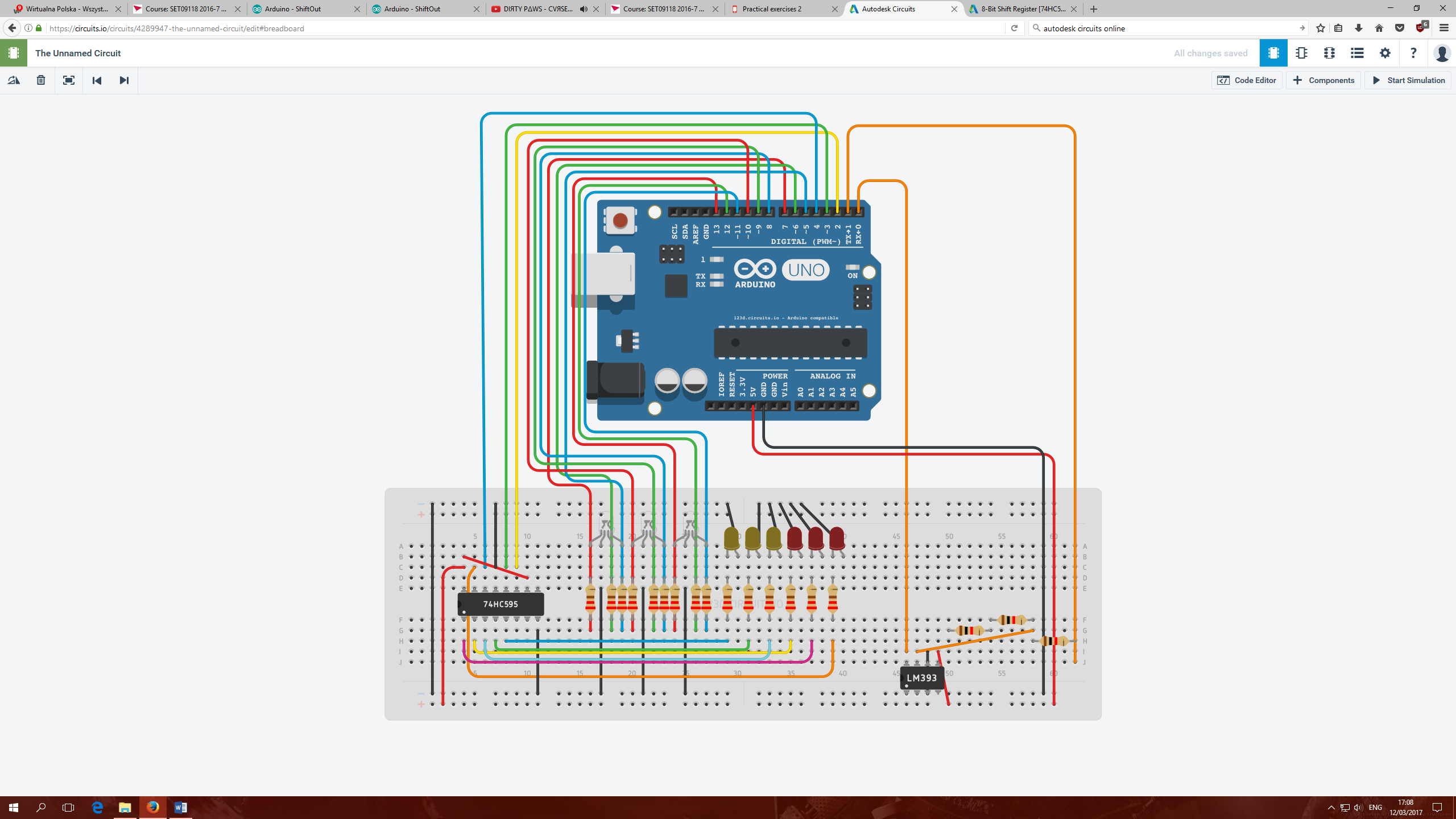
- Processing version 3.3 to create the Android application, it was discovered that processing can handle Arduino connection with proper libraries so that it can handle GUI side, game logics and communication with the Arduino[1]

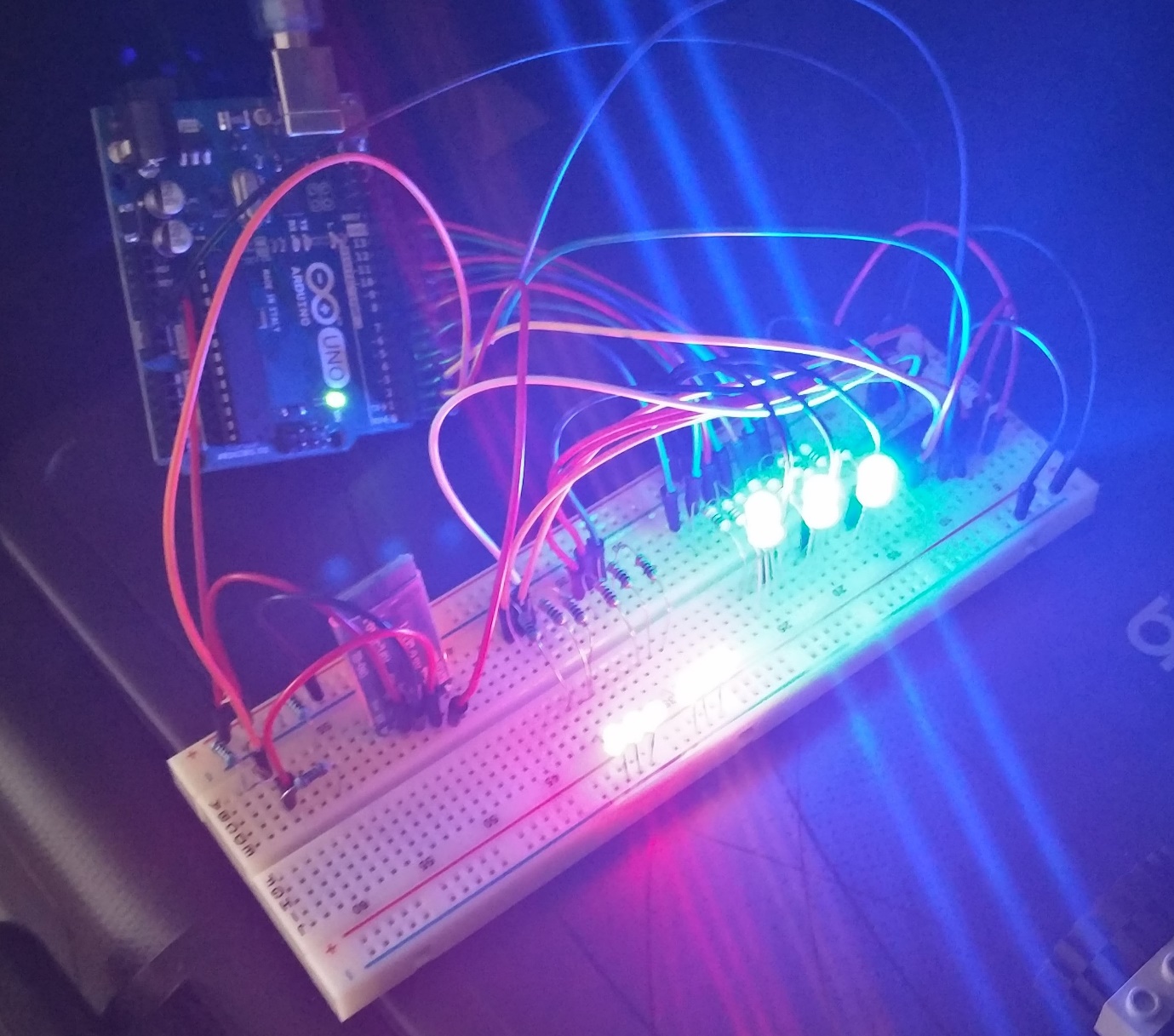
- Processing GUI builder tool – G4Ptool version 4.2

- Processing Bluetooth communication – Ketai library[3]

Outcome of analysis process:  
- Arduino has only 14 pins available and the device would be using 17 outputs so that a solution to this problem is shift register, which will require itself 3 outputs but provides 8 additional pins, 6 of which will be used for red and yellow led lights. This reduced overall design to exactly 14 output pins.

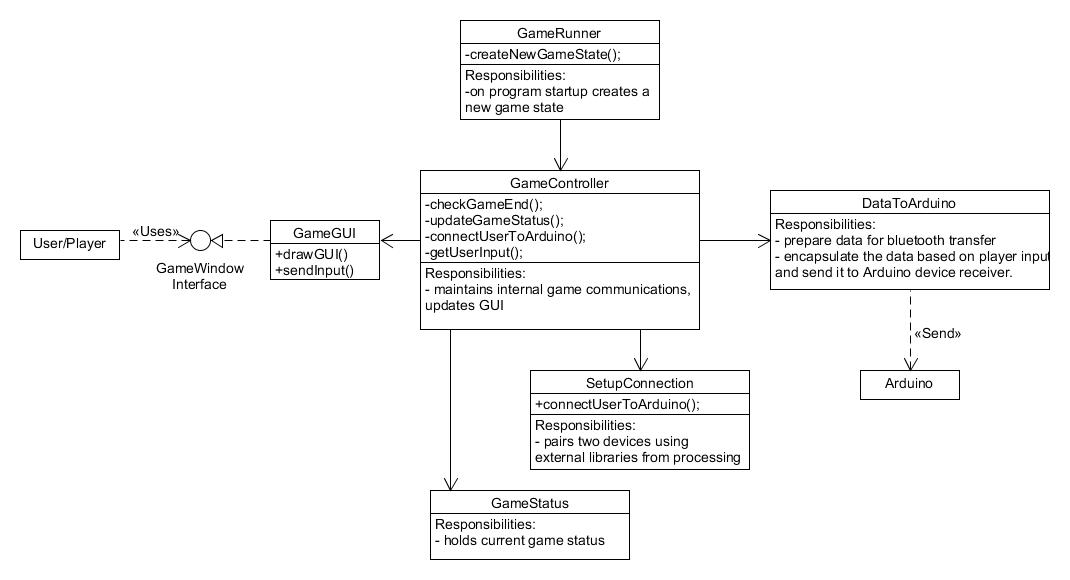
- Since it’s a good practice to draw something first, then create it - the prototype design was first created with the use of Autodesk Circuits[4] and it was shown below:  
(Bluetooth was missing from the list of components and it was replaced with LM393 chip)

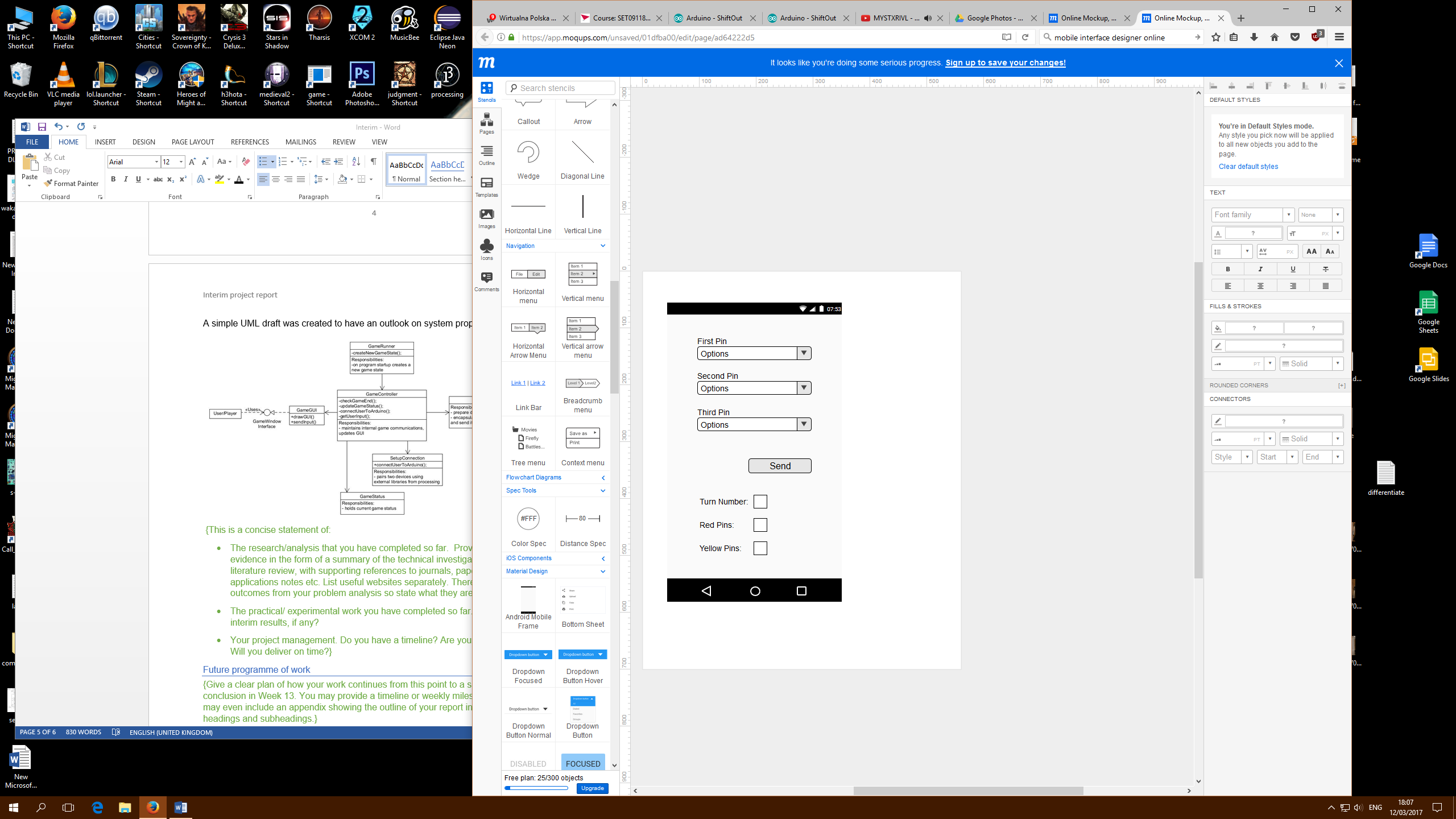


This prototype was then built and tested with basic application lighting all LEDs:  


Additionally Bluetooth was connected but not configured in order to check if Arduino will handle all the items turned on at the same time without any issues.  
  
Given the positive results, the next stage was to create an application draft design which will provide the user ability to interact with the system such as:  
- choose sequence of 3 colours (blue, red, green, white, purple),  
- send processed user input to Arduino to see the results,  
- check game status,   
- manage game start and end

A simple UML draft was created to have an outlook on system proposed behaviour:



And User GUI draft as proposed is:  


The project has a schedule and according to it the student is on track, the report was written before progressing to current week’s next stage - to develop the Android application. The online resources accessed and found are mentioned in bibliography.

Future programme of work

|  |  |  |
| --- | --- | --- |
| Date | Deliverable | Comment |
| 17/02/2017 | Project initial documentation | * Completed |
| 24/02/2017 | Project device scheme, and application GUI draft | * Completed |
| 03/03/2017 | Project initial device and GUI working prototypes | * Completed |
| 10/03/2017 | Catch up work | * Interim report and start of application development |
| 17/03/2017 | Development of the application game logics and first release for testing |  |
| 24/03/2017 | Any other catch up work if needed |  |

Issues and concerns

Currently it is not known yet if Processing GUI building external tool will work as expected, although initial try-out shown positive results.  
A better understanding of how data is sent over Bluetooth must be made so that the application can send a nice single package to Arduino to be read and used.

References

[1]"Arduino Playground - Processing", *Playground.arduino.cc*, 2017. [Online]. Available: http://playground.arduino.cc/Interfacing/Processing. [Accessed: 15- Feb- 2017].

[2]"Arduino - ShiftOut", *Arduino.cc*, 2017. [Online]. Available: https://www.arduino.cc/en/Tutorial/ShiftOut. [Accessed: 12- Mar- 2017].

[3]"Connecting Arduino to Processing - learn.sparkfun.com", *Learn.sparkfun.com*, 2017. [Online]. Available: https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing. [Accessed: 15- Feb- 2017].

[4]"Bring ideas to life with free online Arduino simulator and PCB apps | Autodesk Circuits", *Circuits.io*, 2017. [Online]. Available: https://circuits.io/. [Accessed: 12- Mar- 2017].

[5]"Mastermind (board game)", *En.wikipedia.org*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Mastermind\_(board\_game). [Accessed: 15- Feb- 2017].

[6]P. Goldsborough, "Arduino and Multiplexing", *Thecodeinn.blogspot.co.uk*, 2017. [Online]. Available: http://thecodeinn.blogspot.co.uk/2013/10/arduino-and-multiplexing.html. [Accessed: 15- Feb- 2017].  
[7]"Online Mockup, Wireframe & UI Prototyping Tool · Moqups", *Moqups.com*, 2017. [Online]. Available: https://moqups.com/. [Accessed: 12- Mar- 2017].

[8]"Quarks Place", *Lagers.org.uk*, 2017. [Online]. Available: http://lagers.org.uk/g4ptool/index.html. [Accessed: 12- Mar- 2017].